GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION
SPONSORED PROJECT INITIATION

Date: 9/12/79

Project Title: RIGEL Software Installation and Training Program

Project No: A-2456

Project Director: Ms Beverly S. Rice

Sponsor: NASA: George C. Marshall Space Flight Center

Agreement Period: From 8/22/79 Until Nov 30 10/31/779

Type Agreement: Purchase Order No. H-40909B

Amount: Not to exceed $3,800

Reports Required: Instructional Documentation (manuals)

Sponsor Contact Person (s):

Technical Matters

Mr. Paul D. Belton
ED42
National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama
35812

(205) 453-3633

Contractual Matters
(thru OCA)

Ms. Bertie Hovis or
Mr. James B. Fly
Contracting Officer-AP21M
National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Marshall Space Flight Center, AL.
35812

(205) 453-2260

Defense Priority Rating: None

Assigned to: CSTL/SAD (School/Laboratory)

COPIES TO:

- Project Director
- Division Chief (EES)
- School/Laboratory Director
- Dean/Director--EES
- Accounting Office
- Procurement Office
- Security Coordinator (OCA) ✓
- Reports Coordinator (OCA)
- Library, Technical Reports Section
- EES Information Office
- EES Reports & Procedures
- Project File (OCA)
- Project Code (GTRI)
- Other

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION

Date: December 7, 1979

Project Title: RIGEL Software Installation and Training Program

Project No: A-2456

Project Director: Ms. B.S. Rice

Sponsor: NASA-George C. Marshall Space Flight Center

Effective Termination Date: 11/30/79

Clearance of Accounting Charges: 11/30/79

Grant/Contract Closeout Actions Remaining:

X Final Invoice

_ Final Fiscal Report

_ Final Report of Inventions

_ Govt. Property Inventory & Related Certificate

_ Classified Material Certificate

_ Other _____

Assigned to: CSTL/SAD (School/Laboratory)

COPIES TO:

Project Director
Division Chief (EES)
School/Laboratory Director
Dean/Director–EES
Accounting Office
Procurement Office
Security Coordinator (OCA)
Reports Coordinator (OCA)

Library, Technical Reports Section
EES Information Office
Project File (OCA)
Project Code (GTRI)
Other _____

CA-4 (1/79)

RASTER-SCAN INTELLIGENT GRAPHICS ENGINEERING LANGUAGE

(RIGEL)

USER'S MANUAL

FOR USE ON MARSHALL SPACE FLIGHT CENTER'S
PDP/11-70 COMPUTER SYSTEM

by

BEVERLY S. RICE

CONSTANCE E. RITTER

GEORGIA INSTITUTE OF TECHNOLOGY
ENGINEERING EXPERIMENT STATION
COMPUTER SCIENCE AND TECHNOLOGY LABORATORY
ATLANTA, GEORGIA 30332

CONTENTS

APPENDICES

LIST OF FIGURES

SPECIAL NOTE:

This manual was created as a general document to be used with RIGEL as it is implemented on any number of computer systems. Therefore, a few of the references in it should be modified in reading for the Marshall Space Flight Center's application.

First, the document describes a system with four Conrac units per Ramtek interface. MSFC has three currently, although a fourth could be accommodated at a later date.

Second, the colors described in the text of the document, namely RED, GREEN, BLUE, WHITE, BLACK, MAGENT, CYAN, and YELLOW, are the eight which are normally chosen for the Ramtek 9202. MSFC's application uses RED, GRN, BLU, WHT, BLK, LRED, LGRN, LYEL, and blinking versions of these eight colors (BRED, etc.).

The information contained in Section 10 (Procedure for Creation and Execution of a Display) and in the Appendices is specific to MSFC's application and should be consulted when in doubt.

1.0 INTRODUCTION

Whereas raster-scan graphics has exhibited remarkable advances in recent years in terms of hardware, there has been a surprising lack of parallel development in the realm of user-oriented software interfaces. The Raster-scan Intelligent Graphics Engineering Language (RIGEL) preprocessor and assembler were designed specifically to provide an interface between the user and the machine-level instruction set of the Ramtek RM-9000 Series Graphic Display System. In this manual, the use of four Conrac display screens, of resolution 640 by 256 pixels, tied to a single Ramtek processor, is assumed. The user must be acquainted with FORTRAN, but it is not necessary that he be a programmer by profession. Only minimal training is required for such a person to become conversant in the high-level RIGEL language. The tremendous savings in personnel time required to generate graphics displays, and the fact that such displays do not have to be coded by professional programmers, make RIGEL a worthwhile investment.

The RIGEL graphics language was designed from a human factors point of view, with the main purpose of providing the user with an easy method of programming graphics displays. RIGEL results in several major benefits for the user as compared with programming in machine code:

1. The process of defining a picture is greatly simplified, thus reducing programming time. As can be seen in Figures 1 and 2, a few lines of RIGEL code, similar to FORTRAN and employing words for commands and arguments which are easy to understand and remember, replace many lines of machine code.
2. A program coded in RIGEL provides a much better mental image of what is being drawn than does a program coded

in machine language. In a sense, RIGEL is self-documenting; thus it is much easier for someone else to understand the program (or for the same person to understand it at a later date, after it has been set aside for a while).

3. It is thus much easier to change a picture, because the line to be changed can be easily located. Also, RIGEL automatically handles the "repercussions" of changes which the user would have to implement manually for a machine-coded program. As an example, if we wanted the lines in the example in Figure 1 to be red instead of blue, we could simply change line (4) to read:

```
     VECTOR START = 320,120; END = 320, 180;
1        FOREGD = RED
```

In the machine code (Figure 2), however, we would have to add the value 0924 (red) at one particular spot in the list of numbers and also remember to change the argument flag at * from 8000 to 8002 to indicate the presence of an additional argument. Similarly, if we were to make a change in data by having line (6) read:

```
     TEXT START = 230,70; STRING = "VALVE # 100",
```

we would have to change the word at *** from 3120 to 3130 and then add 3020. Additionally, the data count word at ** would have to change from 000A to 000C.

4. Ease of determining and correcting errors is also enhanced by RIGEL. Failure to indicate the correct number of arguments or data words in the machine code version would cause subsequent instructions to be treated as

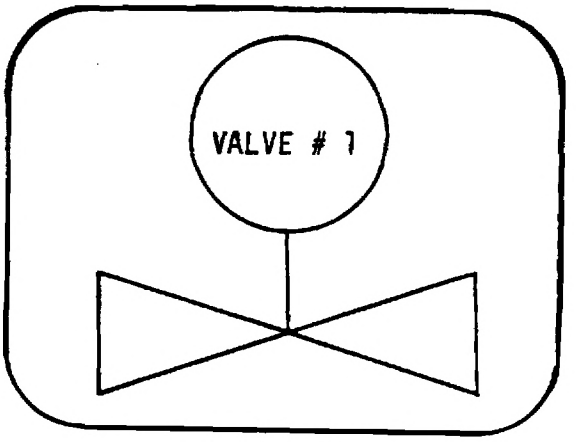
data and data as instructions, resulting in a garbled, meaningless picture. Such an error usually requires a considerable amount of time to trace. With RIGEL, however, instructions and data are generated by the processor. The data may have an error in value caused by a user's incorrect entry, but it will nonetheless be treated as data, thus simplifying the process of correction.

5. RIGEL provides the user with the ability to create and implement customized user symbols.
6. RIGEL also permits the user to decrease picture-generation overhead in time-critical situations. These last two benefits are described elsewhere in this documentation.

It is assumed that the reader of this manual has a working knowledge of the FORTRAN programming language; therefore, details related to programming in these areas are not presented. This manual gives a general description of the various Ramtek instructions and arguments and details of how to implement them in the RIGEL language. It was not the authors' purpose to duplicate the contents of the Ramtek Programming Manual, which the user should consult for specifics of what these commands do. This manual does include, however, items of clarification which the designers of RIGEL, as users of their own product, felt were either omitted or unclear in Ramtek's documentation.

THE CODE NECESSARY TO GENERATE
A SIMPLE PICTURE, WITH THE AID OF RIGEL

Desired Picture:



RIGEL Code:

```
FIXED
PROGRAM VALVE
BEGPIC                                                          (1)
ERASE FOREGD=BLUE; BACKGD=WHITE                                 (2)
SYMBOL CIRCLE SIZE=50; POSX=320; POSY=120                       (3)
VECTOR START=320,120; END=320,180                               (4)
VECTOR START=170,140; END=470,220; END=470,140;                 (5)
1     END=170,220; END=170,140
TEXT START=230,70; STRING='VALVE # 1'                           (6)
ENDPIC
END
```

NOTE: Parenthesized numbers to the right of the code are used to reference corresponding machine code in Figure 2.

Figure 1

THE CODE NECESSARY TO GENERATE
THE SIMPLE PICTURE IN FIGURE 1,
WITHOUT THE AID OF RIGEL

Machine Code:

Column 1	Column 2	Column 3	Column 4	Column 5
(1) 0902	0007	0000	0004	008C
FFFF	0009	0000	0140	00AA
0FFF	0000	(3) 0F03	00AA	00DC
0FFF	0000	9000	(4) 0E03	00AA
0000	0000	0000	8000 *	008C
0000	0000	0001	0140	(6) 0C23
0000	0000	0000	0078	8000
0000	0000	0004	0004	00E6
0000	0000	0000	0140	0046
026C	0000	0000	00B4	000A **
00EF	0000	0000	(5) 0E03	5641
0000	0000	0000	8000	4C56
0000	0000	0000	00AA	4520
027F	0000	FE6F	008C	2320
00EF	(2) 0902	0000	0010	3120 ***
0000	8006	07D0	01D6	
0007	0249	0140	00DC	
0009	0FFF	00AA	01D6	

Figure 2

2.0 RETENTION OF FORTRAN FEATURES

The RIGEL translator examines each statement in the user's input program to see if it is a graphics instruction (a statement valid in RIGEL but not in FORTRAN). If it is recognized as a graphics instruction, the translator generates the corresponding FORTRAN statements. If, however, the statement is not recognized as a graphics instruction, then the translator assumes it must be a FORTRAN statement and passes it through unaltered. Thus, the RIGEL system does not restrict the use of FORTRAN statements; it simply provides a set of additional statements which may be used.

Note in particular that this scheme does not check each statement against an exhaustive list of all possible FORTRAN instructions. To do so would have greatly increased the RIGEL overhead in terms of core storage and execution time and would have duplicated much of the effort of the FORTRAN compiler. In practical terms, this means that a misspelled graphics instruction statement will be passed undetected and unaltered from RIGEL to the FORTRAN compiler, which will then detect and flag the error. This was deemed to be acceptable, since the error is eventually flagged. In all other cases, where the first word of a graphics instruction is spelled correctly, any other errors in the statement will be detected by the RIGEL preprocessor.

3.0 RIGEL: PREPROCESSOR AND GRAPHICS ASSEMBLER

RIGEL functions in two separate parts (see Figure 3) in order to take a user's program written in FORTRAN plus graphics language statements and create a picture from it. The user's source program is first read by the RIGEL Preprocessor (RIGEL), which converts the program to a pure FORTRAN program. At link edit time, the set of RIGEL Graphics Generation Routines, which are responsible for generating the Ramtek object code, is linked to the user's program to form a complete picture program which is then ready for execution.

CREATION OF A PICTURE USING RIGEL

(a two-step process)

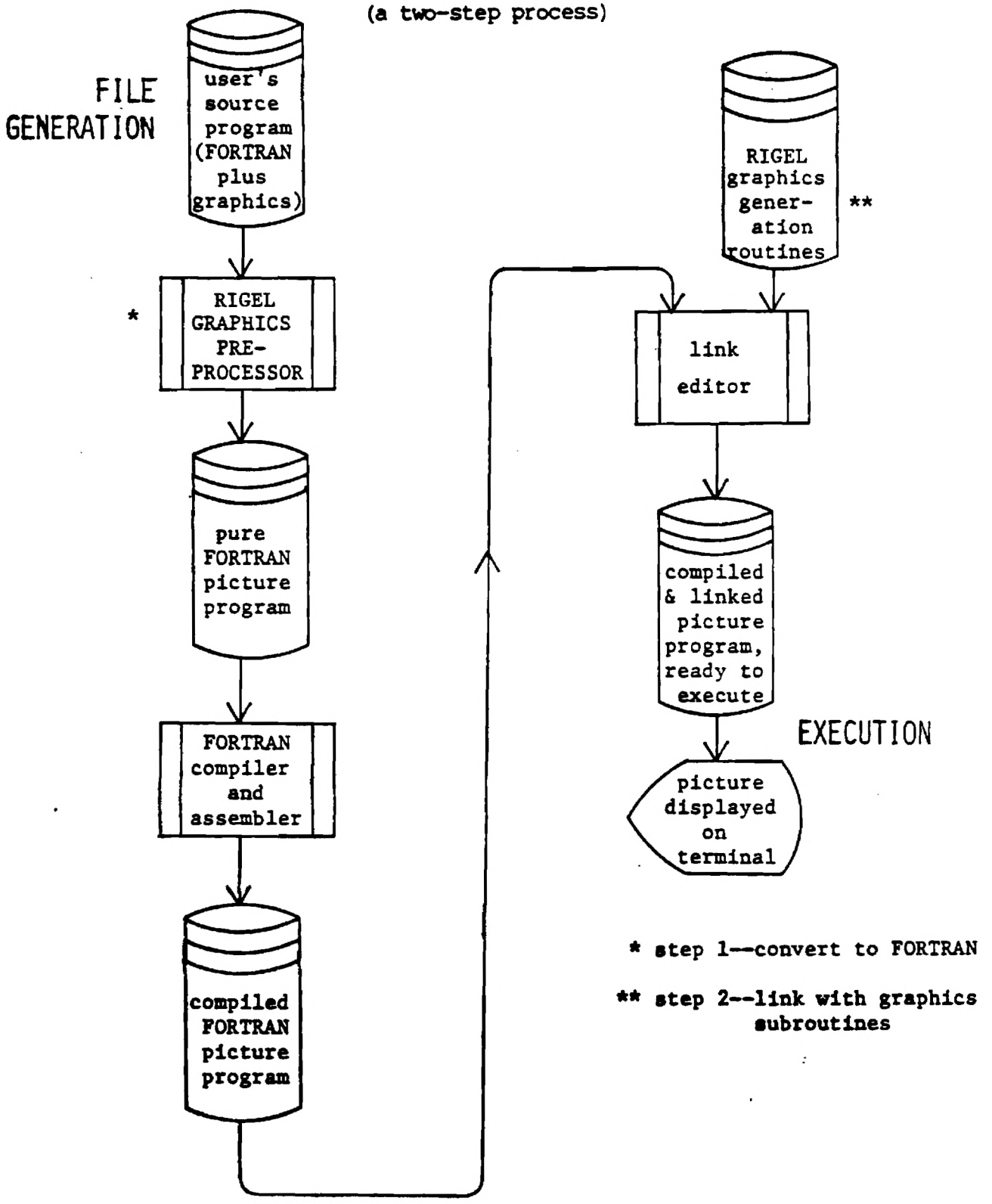


Figure 3

4.0 TIME-CRITICAL APPLICATIONS

Incorporated in RIGEL is an important mechanism for speeding up displays on applications that are time-critical; namely, pictures can be divided into two portions, one called the "fixed-background" portion, and the other called the "real-time" portion. A series of instructions, each of which requires a finite amount of time to execute, must be processed in order to draw a picture. It may require several seconds to interpret all of these instructions. A large portion of the picture, however, may consist of "constants" (legends, scales, etc.) that are independent of real-time data and that do not change throughout the duration of the picture. This portion of the picture can be generated once by executing each instruction in sequence (see Figure 4). The bit-image pattern of the picture can then be read back from the display and stored on disk. From then on, whenever that picture is displayed, the background image can be written directly back to the screen without having to re-execute each separate instruction. Then, only the portion of the picture which is dynamic has to go through the instruction-execution process.

Note that there is a trade-off between the increased speed with which a picture can be displayed by separating it into these two components, and the amount of disk storage space which is required to store the fixed-background portion. Unless a special packing algorithm is applied, a fixed-background bit-image pattern will require 163,840 words for storage (that is, one word for each of 640 x 256 pixels). Packing could reduce this figure to 40,960 words for a picture dedicated to one screen.

DIVISION OF PICTURE INTO FIXED-BACKGROUND AND REAL-TIME PORTIONS

(recommended but not required for time-critical applications)

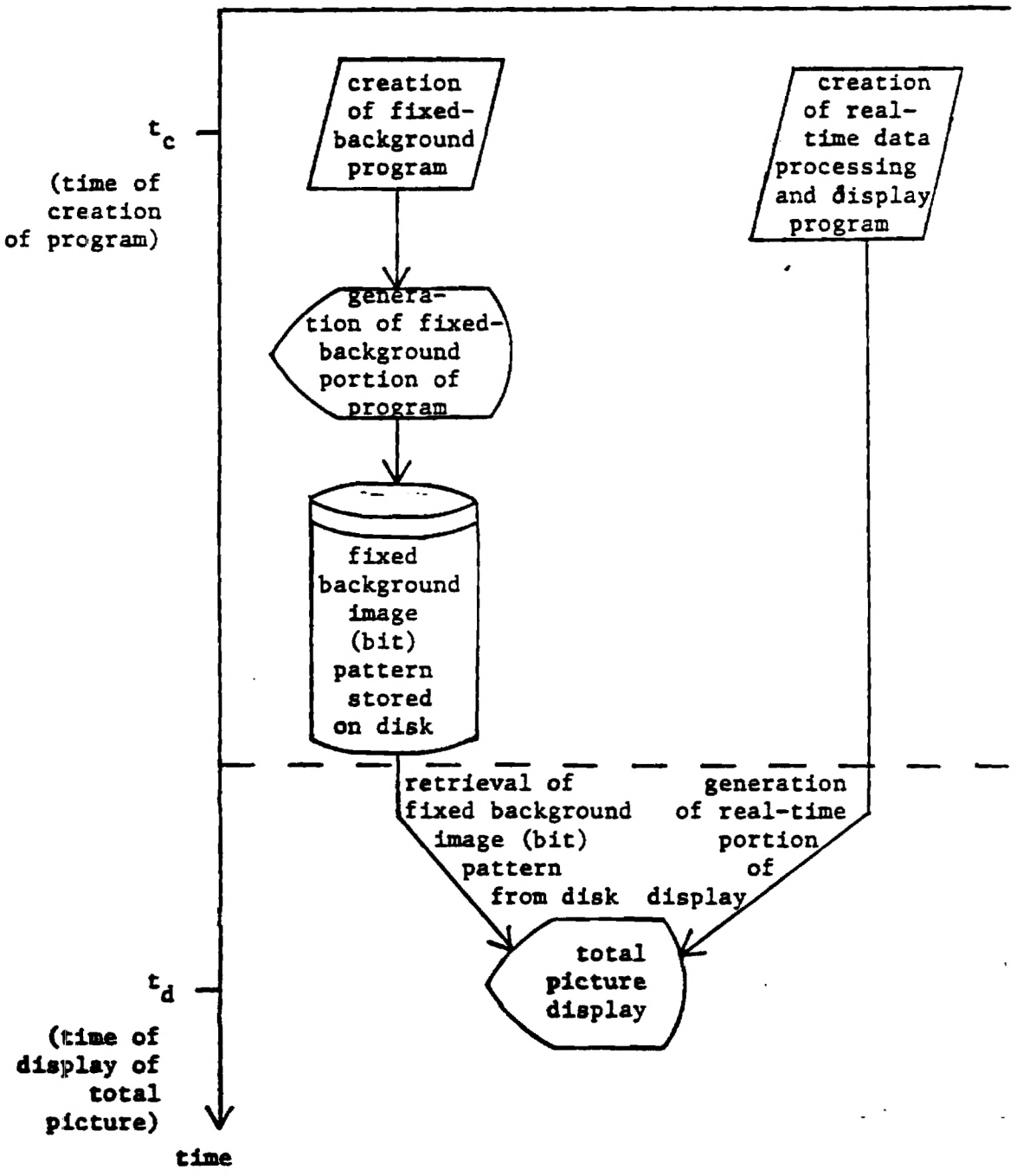
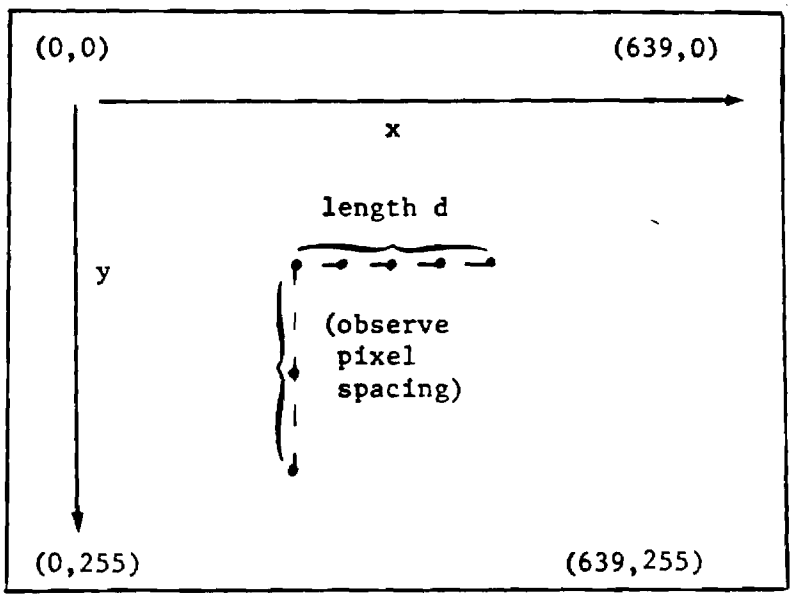


Figure 4

5.0 SCREEN COORDINATE SYSTEM

The CONRAC display screen used in the current application has the following format:



That is, there are 640 pixels (dots) in the horizontal (X) direction and 256 pixels in the vertical (Y) direction. Unlike the normal representation of the Cartesian coordinate system, in which the positive directions are defined to be to the right and upward, the Ramtek system establishes right and downward as positive, thus locating the origin (0,0) in the upper left-hand corner of the screen. Absolute X values can range from 0 to 639, and absolute Y values from 0 to 255 (relative addressing will be described later).

The reader should take careful note that there are twice as many pixels per unit length in the X-direction as there are in the Y-direction. For example, in order to draw a square, if the user selects sides of length 100 pixels in the X-direction, then the sides in the Y-direction must be 50 pixels in length.

6.0 DESCRIPTION OF THE HIGH-LEVEL LANGUAGE

RIGEL is a graphics preprocessor which extends the normal FORTRAN control structures to include graphics commands. RIGEL will accept any statements that the FORTRAN compiler will accept, in addition to those specific to RIGEL. Unlike other graphics packages, RIGEL does not simply consist of a set of subroutines which are callable from the user's program. Instead, RIGEL is a set of key-word commands which can be embedded in the user's program. Although the decision to pursue this approach in the design of RIGEL necessitated the creation of a preprocessor to generate a FORTRAN program, whereas the use of subroutine calls would have resulted in a FORTRAN-compatible program, the former approach was preferred from the standpoint of optimizing ease of use. For example, a subroutine to produce a line of text might have looked like:

```
CALL TEXT (POSX, POSY, ROTATE, ARRAY, NCHARS, SCALE)
```

The arguments in the calling sequence would all have to be specified, in the proper order, in order for the instruction to function correctly. That is,

```
CALL TEXT (POSX, POSY, ARRAY, NCHARS, SCALE)
```

or `CALL TEXT (POSX, POSY, ARRAY, ROTATE, NCHARS, SCALE)`

would have been incorrect. A casual user would have had to program with a manual in hand, a situation which is normally very frustrating for users who are not primarily programmers. In the RIGEL language, however, a handful of arguments with mnemonic names can freely combine with the various mnemonic instructions. The same text instruction in the RIGEL language might have looked like:

```
TEXT POSX = 10; POSY = 50; ARRAY = ICHAR, 15
```

or `TEXT ARRAY = ICHAR, 15; SCALE = 2`

or TEXT POSY = 50; ARRAY = ICHAR, 15; POSX = 10; SCALE = 2 or any number of other possibilities with arguments in any order (or omitted when they assume default, or previous, values).

6.1 FEATURES

The Raster-scan Intelligent Graphics Engineering Language (RIGEL) includes eight control words, eight instructions (six of which require data specific for that instruction), twelve general-purpose Ramtek arguments which may be used with any of the instructions, two modes, and ten symbol arguments, which are used only for items in the user symbol library. Sections 6.2 through 6.7 discuss in detail these various key words and their functions. Appendix A provides a summary of them.

The RIGEL language, which is nearly free-format, encompasses several useful general features:

(1) Like FORTRAN, instructions and arguments do not have to begin in any particular column.

(2) Like FORTRAN, continuation lines are permitted for an instruction, and expressions can span lines.

(3) Spaces can be inserted between (but not in the middle of) variable values for legibility.

(4) Arguments (descriptors) for an instruction can be specified in any order, or omitted entirely if they are to assume default values.

(5) Argument values generally have several alternate forms of specification, to suit different user applications. For example, numeric text data can be entered

as STRING = "6.5"

or ARRAY = IDATA, 3 (where IDATA contains the characters)

or VALUE = X, 3, 2 (where X = 6.5).

(6) Argument values can be specified in the text of the program; e.g.,

```
IF (X .GT. 10.0) FOREGD = RED
```

or `WINDOW(1) = 100`

(7) A special set of symbols can be custom-designed for the user (e.g., STAR) and can then be referenced as though they were part of the normal instruction set. These symbols have their own set of arguments, in addition to being allowed to access any of the standard Ramtek arguments. The above-named symbol could be used by specifying:

```
SYMBOL   STAR   COLOR  =
1     BLUE; SIZE = 10; POSX = 50;
2     POSY =  150
```

6.2 GENERAL PROGRAM STRUCTURE

There are three types of programs recognized by RIGEL: fixed-background display, real-time update display, and user symbol definition programs. Each of these three has a certain program structure which must be observed. One of the three key-words -- "FIXED", "RTIME", or "SYMDEF" -- must be specified on the first line of the program. The rest of the program consists of a mixture of standard FORTRAN and RIGEL graphics instructions.

FIXED and RTIME program structures are identical, except for the first line which identifies the program type. In a FIXED or RTIME program, the command "BEGPIC" ("begin picture") must precede all other graphics instructions, and at least one "ENDPIC" ("end picture") command must precede the FORTRAN "END" statement. Thus, the structure of a FIXED or RTIME program would be:

```
FIXED                          [or RTIME]
. . .
FORTRAN declarations *
other FORTRAN statements
. . .
BEGPIC
. . .
FORTRAN and/or graphics statements
. . .
ENDPIC                    (this statement may be repeated)
. . .
FORTRAN statements        (e.g., END)
```

A SYMDEF program is not in standard program format, but rather consists of the instruction "SYMDEF" followed by one of more sets of instructions defining symbol(s), each of which is delimited by the instructions

"DEFINE symbolname" and "ENDDEF."

The structure of a SYMDEF program defining N symbols would thus be:

```
SYMDEF
DEFINE name1
. . .
FORTRAN declarations *
. . .
FORTRAN and/or graphics statements
. . .
ENDDEF
DEFINE name2
. . .
FORTRAN declarations *
. . .
FORTRAN and/or graphics statements
. . .
ENDDEF
. . .
. . .
DEFINE nameN
. . .
FORTRAN declarations *
. . .
FORTRAN and/or graphics statements
```

. . .

ENDDEF

* NOTE: The FORTRAN declaration section includes those which are required for RIGEL (see Appendix C), as well as any additional declarations which the user may need to make for his particular application.

FURTHER NOTE: A symbol definition program may NOT use another symbol within its body.

6.3 CONTROL STATEMENTS

The eight control statements in the RIGEL language -- FIXED, RTIME, SYMDEF, BEGPIC, ENDPIC, DEFINE, ENDDEF, and SYMBOL -- are not directly related to any one particular Ramtek instruction, but rather perform chores such as initialization and guidance of the RIGEL Preprocessor in translating the user's program. These eight control statements and their functions are described below.

6.3.1 FIXED

This instruction is required to be the first word of a fixed-background display program; it is invalid anywhere else. Among other functions, it causes RIGEL, upon encountering a BEGPIC instruction, to erase and initialize all four of the Conrac display units on one Ramtek interface. It automatically sets the display mask so that the picture which is about to be drawn will appear on all four units. The bit image pattern generated in the Ramtek memory can then be read and stored on disk for future display on any one or more of the Conrac units.

6.3.2 RTIME

This instruction is required to be the first word of a real-time update display program; it is invalid anywhere else. When RIGEL encounters an RTIME instruction, it automatically sets the display mask so that the picture which is about to be drawn will appear on only one Conrac display unit. Furthermore, whatever is already present on the screen is not erased; the new image is simply superimposed over the previous one. This program should not be used to create a bit pattern.

6.3.3 SYMDEF

This instruction is required to be the first word of a symbol definition program; it is invalid anywhere else. Among other functions, it alerts RIGEL

to the fact that symbol subroutines, rather than picture display programs, are being created. These symbol subroutines will later be referenced by SYMBOL statements which are described in Section 6.3.8. Note that more than one symbol can be defined in a SYMDEF program, as shown in Section 6.2.

6.3.4 BEGPIC

This instruction ("begin picture") causes the initialization of a display program. It should therefore appear in the user's FIXED or RTIME program before any other Ramtek instructions are issued, and it should appear only once. BEGPIC is invalid in a SYMDEF program. BEGPIC initializes the values of the various Ramtek arguments and modes and, in the case of a fixed-background bit pattern program, also erases the screen.

6.3.5 ENDPIC

This command may appear any number of times in a FIXED or RTIME program; it is invalid in a SYMDEF program. Each ENDPIC causes the current contents of the Ramtek instruction buffer in core to be written to the Ramtek hardware. At least one ENDPIC is therefore required in a FIXED or RTIME program, and the (final) ENDPIC statement must not be followed by any graphics instructions. In most practical applications, this means that the next program statement will be the FORTRAN "END" statement.

6.3.6 DEFINE

This statement is used in a SYMDEF program to signal the start of a new symbol definition. The name of the symbol follows the word "DEFINE"; e.g.,

```
DEFINE  STAR.
```

This command has no meaning in a FIXED or RTIME program.

6.3.7 ENDDEF

This statement is used in a SYMDEF program to signal the end of a symbol definition. It has no meaning in a FIXED or RTIME program.

6.3.8 SYMBOL

This control statement is used in a FIXED or RTIME program to evoke a symbol that has been previously created in a SYMDEF program. The correct symbol is indicated by the symbol name following the word "SYMBOL", such as SYMBOL STAR. Further details on this instruction are given in Section 7.0. "SYMBOL" is invalid in a SYMDEF program.

6.4 INSTRUCTIONS AND THEIR ASSOCIATED DATA

The RIGEL language includes eight Ramtek instructions -- SET, ERASE, TEXT, RASTER, VECTOR, CONIC, PLOT, and FONT. The section on general program structure, Section 6.2, indicates where these instructions may be used. The general format of the Ramtek instructions is:

```
inst   arg1;   arg2;   ...   argn
```

where "inst" is the mnemonic representation of one of the Ramtek instructions, and "argi" is either an argument, data, or mode value of the form:

```
argi = val1, ... valn
```

where "argi" is the mnemonic representation of one of the key words for arguments, data, or modes. This key word is followed by an equal sign. In cases where an argument requires multiple values, they are separated from each other by commas; arguments (or data or modes) are separated from each other by semi-colons. No semicolon is placed after the last argument. There may be any number of these arguments, in any order, and they may span continuation lines. Since data are obligatory for all instructions except SET and ERASE, these instructions are the only two which may appear without any arguments.

Argument and mode values are optional and may be specified for any of the instructions, so they are described separately (see Sections 6.5 and 6.6); but data values are required and are specific to certain instructions (e.g., STRING may be used with TEXT but not with VECTOR), and so they are described in conjunction with the instruction itself. In most cases where data values are required, there are several representations by which the values may be specified, thus providing the user greater flexibility.

The user should note in the following data descriptions that all arrays are one-dimensional. In cases where the data consists of x, y coordinate points, two separate arrays are used, one for the x-coordinate values and one

for the y-coordinate values.

6.4.1 SET

The SET instruction does not require data, but it may be accompanied by any number of arguments. It sends new values of the specified arguments to the Ramtek hardware, which could affect subsequent instructions, but does not itself cause any change to the representation on the display screen. In that sense, SET is like the no-op instruction which occurs in many programming languages.

NOTE: In the opinion of the authors, the SET instruction is of no value, since it does not cause a screen change. In place of an instruction like:

```
SET FOREGD = BLUE
```

we would use:

```
FOREGD = BLUE.
```

The first instruction forces an output to the Ramtek, including separate words for the instruction. The second merely stores a new value for the FOREGD argument in the RIGEL common. This argument could change again before the next instruction is output, in which case only the new value would be output using our method. Even if it does not change, it goes out as a one-word argument for a meaningful instruction, rather than as a several-word separate instruction.

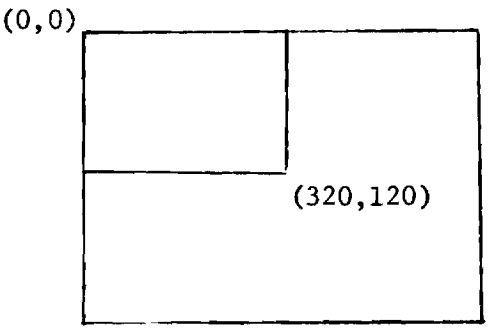
6.4.2 ERASE

The ERASE instruction does not require any data. It sets all of the pixels within the rectangular area defined by the four WINDOW argument values to the BACKGD color (or to the FOREGD color, if reverse background mode (BK = 1) has been selected). In normal background mode, foreground is the color of

lines drawn by VECTOR, characters created by TEXT, etc. These are drawn upon the background color, which is established by first erasing a rectangular area. In reverse background mode, erasing is performed in the foreground color, and lines and characters are generated in the background color. It is advisable for the user to explicitly define a WINDOW with the ERASE instruction. Otherwise, the last WINDOW in effect will be used as a default.

SAMPLE DISPLAY

```
ERASE WINDOW = 0,0,320,120; BACKGD =
1     BLUE
```



6.4.3 TEXT

The TEXT instruction is used to generate ASCII characters on the screen. These characters, and the number of characters, constitute the required data and can be represented in any one of four formats:

```
    TEXT ARRAY = name, length
or  TEXT STRING = "abc"          (or 'abc')
or  TEXT VALUE = xx.x, m, n
or  TEXT VALUEL = xx.x, m, n
```

where: "name" is the name of an integer array containing ASCII characters

packed two per word

"length" is the number of words in array "name"

"abc" is an ASCII character string

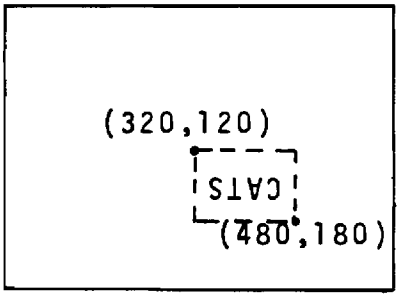
"xx.x" is a real number expression with "m" significant digits (exclusive of sign and decimal point) to be represented and "n" digits to follow the decimal point

note: VALUE specifies a right-justified number (the normal representation); VALUEL specifies a left-justified number

EXAMPLES:
```
TEXT ARRAY = INAME, 20
TEXT STRING = "THIS IS A STRING"
TEXT VALUE = 37.5, 3, 1
TEXT VALUEL = (x+y)/100, N, 2
```

SAMPLE DISPLAY

```
TEXT WINDOW = 320,120,480,180;
1     SCAN = 3; FOREGD = BLACK;
2     STRING = "CATS"; START = 480,180
```



6.4.4 RASTER

The RASTER instruction writes data to the display unit on a bit-per-pixel basis (see Section 3-50 of the Ramtek Programming Manual for further details). RASTER data is specified in an array:

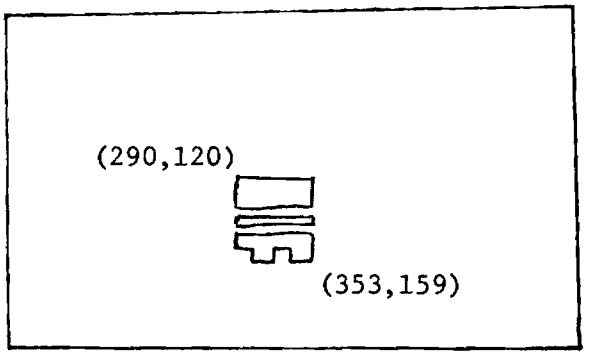
RASTER ARRAY = name, length

where: "name" is the name of an integer array containing Raster data

"length" is the number of words in array "name".

SAMPLE DISPLAY

```
 DIMENSION ISOL(10)
      :
 DATA ISOL(1), ISOL(2), ISOL(3), ISOL(4),
1     ISOL(5), ISOL(6), ISOL(7), ISOL(8),
2     ISOL(9), ISOL(10)/4* "177777, 0,
3     2* "177777, 0, "170360, "7417/
      :
 RASTER FOREGD = RED; WINDOW = 290,
1     120,353,159; SCALE = 22; ARRAY
2     = ISOL, 10
```



6.4.5 VECTOR

The VECTOR instruction is used to draw lines from the point specified by the argument START to successive end points specified by the data, which represent x,y screen coordinates. (Note that if the START argument is not specified, then the Current Operating Point is used as the startpoint; see Section 6.5.12 for details.) VECTOR data can be specified as a set of two arrays or as a series of endpoints:

```
VECTOR ARRAY = name1, name2, length
```

or `VECTOR END = x1,y1; END = x2,y2; ... END = xn,yn`

where: "name1" is an array containing "length" successive x-coordinates

"name2" is an array containing "length" successive y-coordinates

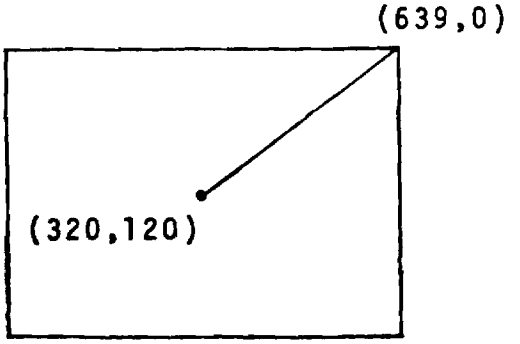
"xi,yi" are the rectangular coordinates of the ith endpoint

EXAMPLE:

```
VECTOR START = ICX, ICY; END = IX, IY; END = 10,50;
  1   END = (10 + 1/2 * J) * 3, K
```

SAMPLE DISPLAY

```
VECTOR  FOREGD = BLUE; START =
    320,120; END = 639,0
```



Note: If, after indexing, any of the START or END coordinate values exceed the screen resolution, the result may be indeterminate.

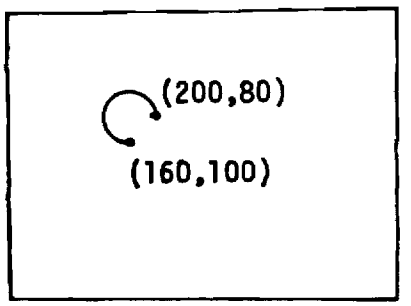
6.4.6 CONIC

The CONIC instruction is used to generate conic sections (circles, parabolas, ellipses, and hyperbolas). The instruction begins at the point defined by START and generates the conic section in a clockwise manner. The data for this instruction represents x, y endpoint screen coordinates. If the conic being drawn passes through the first endpoint, the conic will be terminated at that point. (If more than one endpoint is specified, all are ignored except the first one.) This means that, in order to draw a complete circle, START x,y coordinates should equal END x,y coordinates. These x,y data values have the same representations as for the VECTOR data in Section 6.4.5; that is,

```
CONIC ARRAY = name1, name2, length
    (Note: Length is probably 1.)
or  CONIC END = x1, y1
```

SAMPLE DISPLAY

```
CONIC FOREGD = BLUE; START = 160,100;
1    END = 200,80; COEFF = 1,4,0,0,
2    -160,0
```



Note: If, after indexing, any START or END coordinate values exceed the screen resolution, the result may be indeterminate.

6.4.7 PLOT

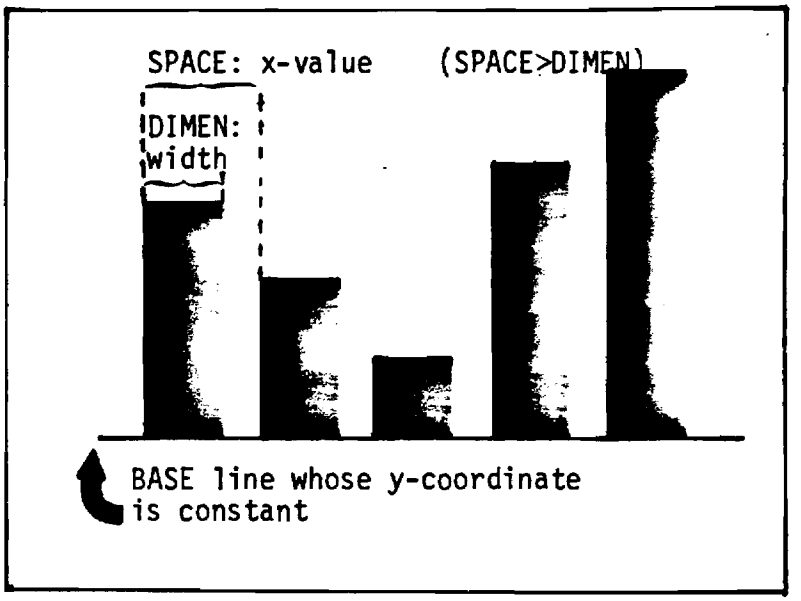
The PLOT instruction is used to draw either filled plots or line plots from the point specified by the argument START to successive points specified by the data, with the increment in either the x or y direction held constant. Plot data specifies either x or y coordinates, but not both. When BASE = 0, SCAN = 0 specifies horizontal line plots and SCAN = 4 specifies vertical line plots. If SCAN = 0 and BASE ≠ 0, the BASE argument value is the constant y-coordinate and the data represents successive x-coordinates whose spacing is determined by SPACE. If SCAN = 4 and BASE ≠ 0, then the BASE argument value is the constant x-coordinate and the data represents successive y-coordinates whose spacing is determined by SPACE. The width of each PLOT segment is determined by DIMEN. The two filled PLOT cases are represented in Figure 5. The format for PLOT data is:

PLOT POINT = P1, P2, ... Pn

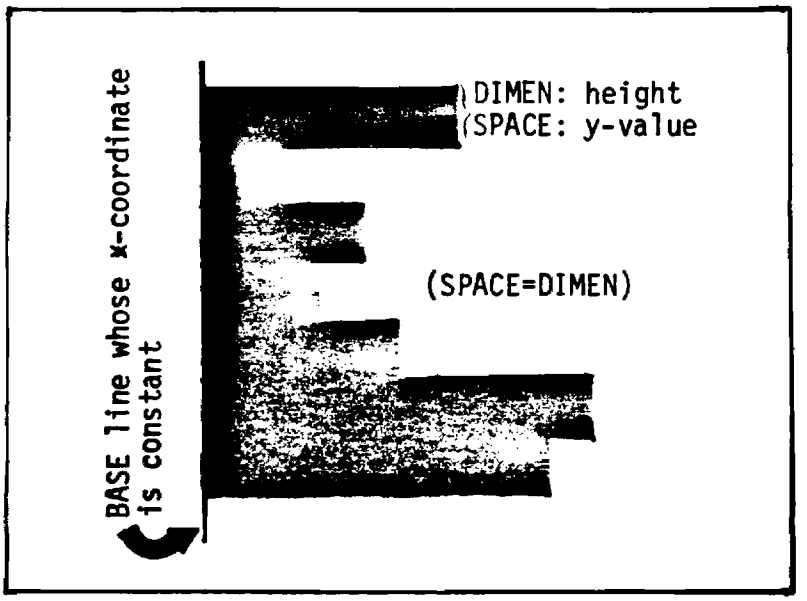
or PLOT ARRAY = name, length

where: "Pi" is the coordinate of the ith endpoint

HORIZONTAL AND VERTICAL FILLED PLOTS



HORIZONTAL PLOT
(SCAN=0)

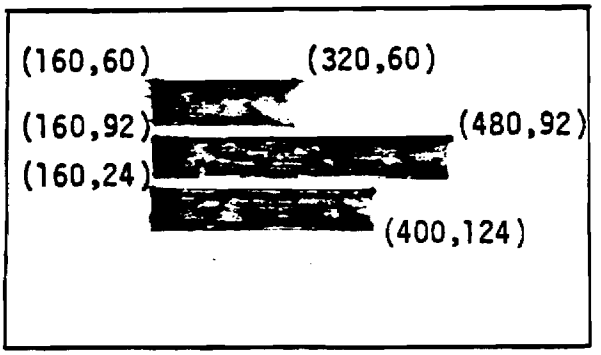


VERTICAL PLOT
(SCAN=4)

Figure 5

SAMPLE DISPLAY

```
PLOT START = 160,60; BASE = 160; DIMEN = 20,30;
1    SPACE = 22,32; POINT = 320,480,400
```



Note: If, after indexing, any POINT coordinate values exceed the screen resolution, the result may be indeterminate.

6.4.8 FONT

The FONT instruction is used to define a character which can later be referenced by the TEXT instruction, and thus displayed on the screen. Each word of FONT data represents two successive lines of pixels for the character, whose dimensions are 8 x 12 pixels. The user should thus specify a data array

six words long. He must also specify a character code for this symbol, so that he can later reference it. Section 3.60 of the Ramtek Programming Manual contains an example of how to describe a character. The format for the FONT instruction is:

FONT CODE = n; ARRAY = name, length

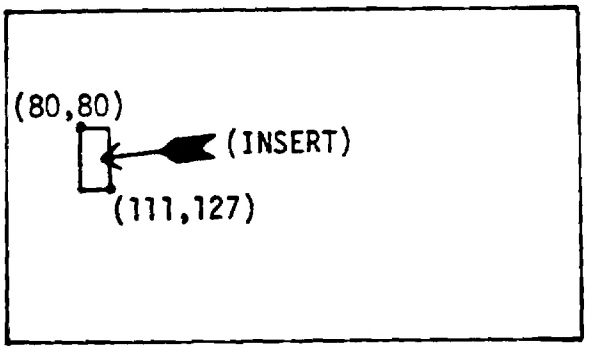
where: "n" represents an octal character code between 40 and 237

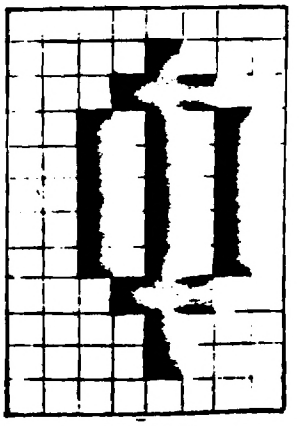
"name" is the name of the integer data array

"length" is the number of words in array "name"

SAMPLE DISPLAY

```
      DIMENSION IPHI(6), IBLK(6), IVAL(1)
          :
      DATA IPHI(1), IPHI(2), IPHI(3), IPHI(4),
     1    IPHI(5), IPHI(6)/"4000, "25034'
     2    "25052, "25052, "4034, "10/
      DATA IBLK(1), IBLK(2), IBLK(3), IBLK(4),
     1    IBLK(5), IBLK(6)/6*0/
          :
      IVAL(1) = 100*256+101
      FONT CODE = 100; ARRAY = IPHI,6
      FONT CODE = 101; ARRAY = IBLK,6
      TEXT START = 80,80; SPACE = 15,18; SCALE =
     1    22; ARRAY = IVAL,1; FOREGD = BLACK
```





INSERT

6.5 ARGUMENTS

The twelve valid RIGEL arguments are listed here and in Appendix A: FOREGD, BACKGD, IX1, IX2, WINDOW, SCAN, DIMEN, SPACE, SCALE, COEFF, BASE, and START. Argument values may be specified for any of the instructions. If an argument value is set in a FORTRAN statement, it affects the execution of the next RIGEL instruction. Otherwise, an argument value is specified in a RIGEL statement and the specified argument value assignment occurs before the instruction is executed.

An argument value assignment is indicated in the following format:

argument name = value(s)

in which the values are separated by commas. Argument value specifications in the same RIGEL statement are separated by semi-colons. When a RIGEL instruction is executed, the relevant arguments are interpreted using the current argument values; the current argument values are the last values specified, or the default values if no specification has been made.

NOTE: When a multi-valued argument is used with a graphics instruction, only the argument name is given, followed by the values, as for example:

```
ERASE WINDOW = 0,0,640,240
```

When such arguments are used in the body of the FORTRAN program as variables, multi-valued arguments are treated as arrays. Done this way, the previous statement would appear as:

```
WINDOW (1) = 0
WINDOW (2) = 0
WINDOW (3) = 640
WINDOW (4) = 240
ERASE . . . .
```

Descriptions of the individual arguments follow.

6.5.1 FOREGD

In normal background (ie, BK=0), foreground is the color of lines drawn by VECTOR, characters generated by TEXT, etc. These are drawn upon the background color, which is established by first erasing a rectangular area. In reverse background mode (ie, BK=1), erasing is performed in the foreground color, and lines and characters are generated in the background color.

The standard foreground argument values are: RED, GREEN, BLUE, WHITE, BLACK, MAGENT, CYAN, and YELLOW. The foreground argument specification is of the form, FOREGD = color. The default foreground argument value is WHITE. Under normal conditions, the user might employ the FOREGD argument with the SET, TEXT, RASTER, VECTOR, CONIC, and PLOT instructions.

6.5.2 BACKGD

For a description of the function of the background argument, see Section 6.5.1 (FOREGD).

The standard background argument values are: RED, GREEN, BLUE, WHITE, BLACK, MAGENT, CYAN, and YELLOW. The background parameter specification is of the form, BACKGD = color. The default background value is BLACK. Under normal conditions, the BACKGD argument is used mainly with the ERASE instruction, in which it defines the color to which the area defined by WINDOW will be erased.

6.5.3 IX1

The Index-1 argument is only used whenever Index-1 addressing is specified, i.e. when IX = 1, but the IX1 values can be set using any addressing mode. The IX1 argument has a horizontal displacement value and a vertical displacement value, e.g. IX1 = x,y. The horizontal displacement value offsets the x-coordinate of a received x,y coordinate pair and the vertical displacement value offsets the y-coordinate of a received x,y

coordinate pair. The IX1 X-Address value can be any integer value from -639 to +639, inclusive, and the Y-Address value can be any integer value from -255 to +255, inclusive.

The actual Index-1 argument values depend upon the received IX1 values and the addressing mode. If the addressing mode is absolute (IX = 0), the received IX1 values are used directly as the actual IX1 values. If the addressing mode is Index-1 (IX = 1), the received IX1 values are summed with the current IX1 values to obtain the new IX1 values. For the Index-2 addressing mode (IX = 2), the received IX1 values are added to the current IX2 values to obtain the new IX1 values. Lastly, for the relative addressing mode (IX = 3), the received IX1 values are added to the current XCOP and YCOP values to form new IX1 values.

The default IX1 values are 0,0.

The IX1 argument could be used with any instruction except FONT. An example of the use of IX1 is shown in Section 6.6.1.

6.5.4 IX2

The Index-2 argument is only used whenever Index-2 addressing is specified, i.e. when IX=2, but the IX2 values can be set using any addressing mode. The IX2 argument has a horizontal displacement value and a vertical displacement value, e.g. IX2 = x,y. The horizontal displacement value offsets the X-coordinate of a received x,y coordinate pair and the vertical displacement value offsets the Y-coordinate of a received x,y coordinate pair. The IX2 X-Address value can be any integer value from -639 to +639, inclusive, and the Y-Address value can be any value from -255 to +255, inclusive.

The actual Index-2 argument values depend upon the received IX2 values and the addressing mode. If the addressing mode is absolute (IX = 0), the received IX2 values are used directly as the actual IX2 values. If the

addressing mode is Index-1 (IX = 1), the received IX2 values are summed with the current IX1 values to obtain the new IX2 values. For the Index-2 addressing mode (IX = 2), the received IX2 values are added to the current IX2 values to obtain the new IX2 values. Finally, for the relative addressing mode (IX = 3), the received IX2 values are added to the current XCOP and YCOP values to form new IX2 values.

The default IX2 values are 0,0.

The IX2 argument could be used with any instruction except FONT. An example of the use of IX2 is shown in Section 6.6.1.

6.5.5 WINDOW

The WINDOW argument is used with the following instructions: SET, ERASE, TEXT, and RASTER. The WINDOW parameter values define an area whose edges constitute the boundaries of a rectangular region that may be accessed by these instructions.

There are four WINDOW argument values which are specified in the following manner: WINDOW = Xl, Yt, Xr, Yb, in which "Xl" is the window's left-most X-coordinate, "Yt" is the top-most Y-coordinate, "Xr" is the right-most X-coordinate, and "Yb" is the bottom-most Y-coordinate. The following restrictions apply to the WINDOW argument values:

(1) $Xl \leq Xr$

(2) $Yt \leq Yb$

(3) $-639 \leq Xl \leq +639$

$-639 \leq Xr \leq +639$

(4) $-255 \leq Yt \leq +255$

$-255 \leq Yb \leq +255$

If, after indexing, any of the WINDOW parameter values exceeds the screen resolution, the result will be indeterminate. The default WINDOW values are

0, 0, 639, 255 such that the entire display screen is within the WINDOW.

Whenever WINDOW argument values are given, the Current Operating Point (COP) is set to the coordinate defined by the table in Appendix B, Part I.

6.5.6 SCAN

There are eight possible scan sequences for the RASTER, TEXT, and PLOT instructions. The particular SCAN sequence is specified in the following manner: SCAN = x where "x" is an integer expression with a value between zero and seven. The default value is zero. The SCAN directions for the RASTER instruction are listed in the Table in Appendix B, Part II, and the directions for the TEXT instruction are listed in Appendix B, Part III. Primary scan is the direction of consecutive pixels. Secondary scan is the wrap-around direction upon reaching a window boundary. That is, when the primary scan completes a line of pixels and is ready for wrap-around, the secondary scan will determine whether the second line of pixels is above, below, to the right or to the left of the first line. The influence of the SCAN argument value upon the RASTER, TEXT, and PLOT instructions is described in Sections 6.4 and 6.7 of this manual.

6.5.7 DIMEN

The DIMEN argument may be used with three instructions: SET, TEXT, and PLOT. The DIMEN parameter defines the dimensions of the alphanumeric font in terms of width and height, or the width or height of individual plot segments in terms of elements or lines. The DIMEN argument has two values, the first of which specifies character width, or plot segment width for horizontal plots, and the second of which specifies character height, or plot segment height for vertical plots. These values are set in a RIGEL program in the following way: DIMEN = x,y, where "x" is an integer expression with a value between 0 and 639, inclusive, and "y" is an integer expression with a value

between 0 and 255, inclusive. The default DIMEN width value is 7 and the default DIMEN height value is 9. When this argument is used to specify character width and height, it is independent of character scale and orientation.

6.5.8 SPACE

The SPACE argument may be used with three instructions: SET, TEXT, and PLOT. This argument has two values, an X-displacement and a Y-displacement, which have default values of 7 and 9, respectively. The SPACE values may be set in a RIGEL program in an argument assignment of the following form: SPACE = x,y, where "x" is an integer expression with a value between -639 and 639, inclusive, and "y" is an integer expression with a value between -255 and 255, inclusive.

With the TEXT instruction, SPACE determines the distance between successive characters and the distance between successive lines. With the PLOT instruction, SPACE determines the increment from plot entity to plot entity, along the plot axis. For more detailed descriptions of the interaction of these instructions with the SPACE parameter, refer to Section 6.7 of this manual.

6.5.9 SCALE

The SCALE parameter determines the number of picture elements to be displayed per pixel element that is received, along the Y-Axis and along the X-Axis. The SCALE argument has seventeen possible values, which are listed in the table in Appendix B, Part IV. A SCALE argument assignment in a RIGEL program would be of the following form: SCALE = x, where "x" is one of the seventeen integer values listed in the table. The default SCALE value is zero, which specifies a 1:1 ratio along the Y-Axis and a 1:1 ratio along the X-Axis.

The table lists for each SCALE value the Y-SCALE ratio of displayed picture elements to received picture elements and the X-SCALE ratio of displayed picture elements to received picture elements.

The SCALE parameter influences the SET, TEXT, and RASTER instructions. Concerning the TEXT instruction, X-SCALE refers to character width and Y-SCALE refers to character height.

6.5.10 COEFF

The CONIC EQUATION COEFFICIENTS argument sets the values of the coefficients which are used by the CONIC instruction. The CONIC instruction can be used to draw circles, ellipses, parabolas, and hyperbolas; however, only the use of the CONIC instruction to draw circles will be discussed here.

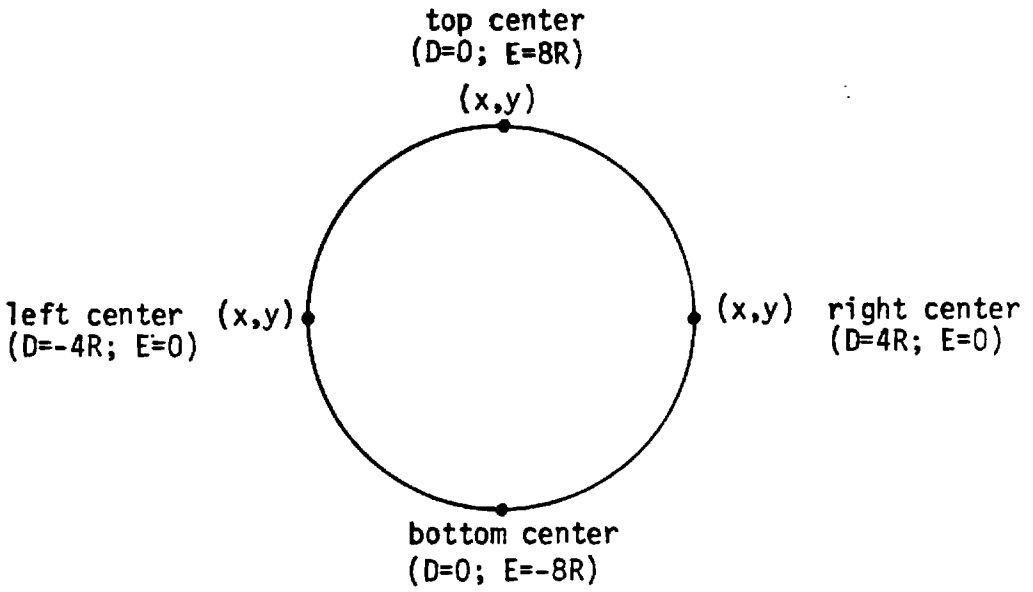
The form of the COEFFICIENTS argument assignment is:

COEFF = A, B, C, D, E, K,

where A = 1, B = 4, C = 0, AND K = the total number of picture elements to be displayed. If K = 0 is specified, K assumes the default value of 1280. The values of D and E depend upon which START point of the circle is used. (See Figure 6). If the bottom center of the circle is used as the START point, then D = 0 and E = -8 * vertical radius. If the left center of the circle is used as the START point, then D = -4 * vertical radius and E = 0. With the START point of the circle at the top center, D = 0 and E = +8 * vertical radius, and with the START point of the circle at the right center, D = +4 * vertical radius and E = 0. The Ramtek-9200 draws each of these circles in a clockwise pattern. Additionally, note that the horizontal radius = 2 * vertical radius. See Section 6.7.4 for additional information.

The default values for A, B, C, D, E, and K are zeros.

GENERATION OF A CIRCLE USING THE CONIC INSTRUCTION



A = 1

B = 4

C = 0

K = number of picture elements
(default = 1280)

R = the vertical radius

Figure 6

6.5.11 BASE

The BASE parameter influences the PLOT instruction in that it specifies whether a filled plot or a line plot is to be drawn. A line plot is a plot in which the POINT value along one plot segment becomes the START point value for the next plot segment. A BASE value of zero specifies a line plot.

If the BASE argument has a non-zero value, then the BASE value is the START point for each plot segment. If the SCAN argument value is between zero and three, then the BASE argument defines the horizontal axis to which the filled-plot segments will be drawn, i.e., BASE specifies a Y-coordinate. Likewise, if the SCAN value is between four and seven, BASE defines the vertical axis to which the filled-plot segments will be drawn, i.e., BASE specifies an X-coordinate.

The BASE argument value is set in a RIGEL program in the following manner: BASE = x, where "x" is an integer expression with a value between -639 and 639, inclusive. The default BASE value is zero.

6.5.12 START

The START argument defines a start-point, which can be specified in the following manner: START = x,y, where "x" is an integer value between -639 and 639, inclusive, and "y" is an integer value between -255 and 255, inclusive. The default START value is 0,0. Note: If, after indexing, either startpoint coordinate value exceeds the screen resolution, the result will be indeterminate.

The instructions which are affected by the START argument are: SET, TEXT, RASTER, VECTOR, CONIC, and PLOT. When the WINDOW or SCAN arguments are specified for the TEXT or RASTER instructions, the appropriate startpoint is automatically calculated; therefore, the START argument need not be specified unless a different startpoint is required. For proper instruction execution,

the START value should be within the WINDOW region. Refer to the table in Appendix B, Part V, for the START default values based on the WINDOW and SCAN values.

If the START argument is not specified for the SET, VECTOR, CONIC, or PLOT instructions, then the current Current Operating Point is used as the startpoint. The START parameter sets the value of the Current Operating Point to the value of the properly indexed startpoint.

6.6 MODES

There are two modes in the RIGEL Language that can be user-controlled. These are the addressing and the reverse background modes. Mode values can be specified in RIGEL instructions or in regular FORTRAN statements.

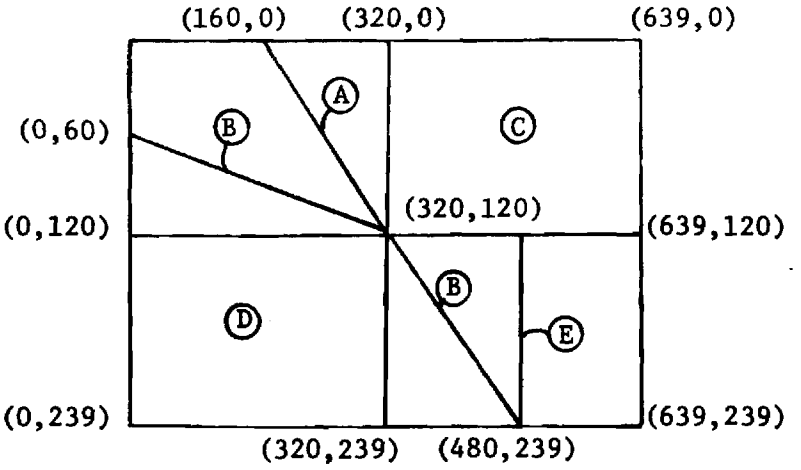
6.6.1 IX

The addressing mode value can be specified in the following manner: IX = n, where n is an integer expression of value 0, 1, 2, or 3. Absolute addressing is indicated by a value of zero, indexed addressing by a value of one or two; and relative addressing by a value of three. The coordinate values that may be affected by the addressing mode are: WINDOW, IX1, IX2, START, POINT, and END.

In the absolute addressing mode, which is the default mode, the x,y coordinate values for the Ramtek and data arguments listed above are used directly as screen coordinates. In the indexed addressing mode, the x,y coordinate values are added to the index values selected to determine the coordinate points. If IX = 1 is specified, the x,y coordinate values are summed (two's complement addition) with the current INDEX-1 specified values in order to derive the effective address. If IX = 2 is specified, the x,y coordinate values are summed (two's complement addition) with the current INDEX-2 specified values in order to determine the effective address. In the relative addressing mode, the x,y coordinate values are added to the Current Operating Point (last screen coordinate read or written) to determine the screen coordinates.

SAMPLE DISPLAY

PORTION OF DISPLAY AFFECTED	RIGEL CODE	TYPE OF ADDRESSING	EFFECT ON SCREEN
A	VECTOR FOREGD = BLACK; 1 START = 160,0; 2 END = 320,120	ABSOLUTE	LINE DRAWN FROM (160,0) TO (320,120) COP SET TO (320,120)
B	VECTOR IX = 3; START = 1 -320,-60; END = 2 320,60; END = 160,119	RELATIVE	LINE DRAWN FROM (0,60) TO (320,120) TO (480,239) COP SET TO (480,239)
NONE	SET IX1 = -160,0; 1 IX2 = 0,-139	RELATIVE	NONE; COP REMAINS AT (480,239) IX1 ASSUMES VALUE OF (320,239) IX2 ASSUMES VALUE OF (480,100)
C	ERASE IX = 2; 1 BACKGD = BLUE; 2 WINDOW = -160, -100, 3 159,20	INDEX-2	ERASED WINDOW: (320,0, 639,120)
D	ERASE IX = 1; 1 BACKGD = RED; 2 WINDOW = -320,-119,0,0	INDEX-1	ERASED WINDOW: (0,120, 320,239)
E	VECTOR IX = 0; 1 START = 480,120; 2 END = 480,239	ABSOLUTE	LINE DRAWN FROM (480,120) TO (480,239) COP SET TO (480,239)



6.6.2 BK

The reverse background mode can be specified by the following method: BK = n, where n is an integer expression of value 0 or 1. Normal background mode, which is the default mode, is specified by a value of zero and reverse background mode by a value of one.

When reverse background mode is indicated, the roles of the foreground (FOREGD) and background (BACKGD) arguments are reversed. For example, the ERASE instruction sets the rectangular area specified by the WINDOW argument to the BACKGD color if normal background mode is indicated, and to the FOREGD color if reverse background mode is selected. Each of the instructions is affected by the reverse background mode.

6.7 INTERACTION OF ARGUMENTS

The twelve valid RIGEL arguments interact with each other, with the instructions, and with the modes. This section provides information in addition to the documentation on the individual arguments, instructions, and modes.

6.7.1 INTERACTIONS CONCERNING TEXT

The WINDOW argument defines the rectangular area into which TEXT will be written. If any pixel of a character is within this area, the entire character will be displayed, even though the entire character may not be within the WINDOW region.

The actual size of a character is determined by the DIMENSION parameter. The maximum values of the DIMENSION argument, with reference to the TEXT instruction, should be 8 pixels wide by 12 pixels high. If the DIMENSION values are less than 7 by 9 (for regular characters) or less than 8 by 12 (for FONT-defined characters), the right-most and bottom-most pixels of the character will not be displayed.

The SPACE parameter determines the distance between the automatic startpoint of a character and the automatic startpoint of the next character. The automatic startpoint of a character is the upper left corner of the character, in whichever direction the character is oriented. Refer to Appendix B, Part VI for the appropriate SPACE values to use with the different SCAN values. When the negative sign is indicated before the SPACE value, in the table, it should also be indicated in the RIGEL program.

The primary and secondary update directions are defined by the SCAN parameter and are listed in the table in Appendix B, Part III. Additionally, the SCAN values determine the startpoint for the TEXT instruction, when the

START argument is not explicitly set. This interaction between SCAN and TEXT is stated in the chart in Appendix B, Part V.

The effect of the SCALE argument upon the TEXT instruction is defined in the table in Appendix B, Part IV.

6.7.2 INTERACTIONS CONCERNING RASTER

The WINDOW parameter defines the exact region into which the RASTER instruction writes FOREGD or BACKGD data, depending upon the BK mode value. The RASTER instruction interprets the data on a bit per pixel basis, i.e. each bit determines whether a pixel gets a value or not. A "one" bit causes the respective pixel to receive a FOREGD or BACKGD value and a "zero" bit causes the respective pixel to receive no value.

The appropriate WINDOW values depend upon the SCALE value of the RASTER instruction. For example, to calculate the WINDOW values for a RASTER with SCALE = 514 which is a ratio of 4 displayed pixels per 1 received bit, the number of element pixels should be 4 times the number of element data bits and the number of line pixels should be 4 times the number of line data bits. If the RASTER mentioned above were to be described using 16 element bits by 10 line bits, the WINDOW should be 64 pixels wide by 40 pixels high. An example set of coordinates would be: WINDOW = 140, 120, 203, 159.

When a scaling-down operation is specified by the SCALE parameter in the horizontal or vertical direction, bits 7, 5, 3, and 1 are used for a ratio of 1:2 and bits 7 and 3 are used for a ratio of 1:4 (the bits are numbered with 7 on the left and 0 on the right).

The START argument does not need to be specified. However, if it is specified, it should be set to the value of the Current Operating Point after the WINDOW setting. For example, with SCAN = 0, START = X1, Yt of the WINDOW region and with SCAN = 1, START = Xr, Yt of the WINDOW region. The

appropriate values are defined in the table in Appendix B, Part I.

The SCAN parameter also defines the primary and secondary RASTER update directions as shown in the table in Appendix B, Part II.

6.7.3 INTERACTIONS CONCERNING VECTOR

If the START argument is not explicitly set in the RIGEL program, the Current Operating Point is used as the startpoint. If more than one endpoint is specified, then continuous, straight lines will be drawn between the consecutive endpoints.

If, after indexing, any of the START or END coordinate values exceed the screen resolution, the result may be indeterminate.

6.7.4 INTERACTIONS CONCERNING CONIC

The START value for the CONIC instruction, used to draw a circle, should be at either the bottom-center, the left-center, the top-center, or the right-center of the circle (see Figure 6). The CONIC instruction draws circles in a clockwise pattern. Two methods may be used to terminate a CONIC plot. First, the number of pixels to be displayed may be specified. Second, an END value may be specified such that only a portion of the CONIC is drawn or, if the END value is equal to the START value, the entire CONIC is drawn.

NOTE; if, after indexing, any START or END value exceeds the screen resolution, the result may be indeterminate.

Refer to Section 6.5.10 for information concerning the COEFF argument.

6.7.5 INTERACTIONS CONCERNING PLOT

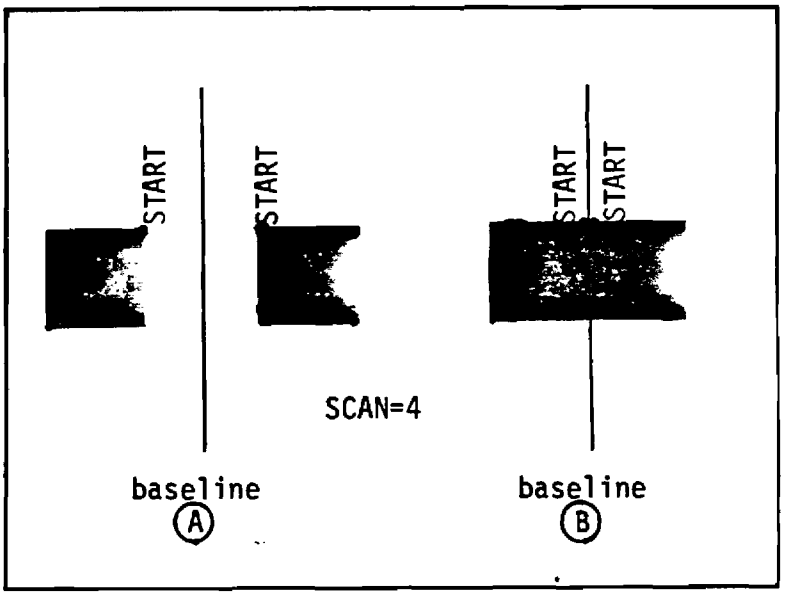
The SCAN argument determines whether the PLOT is horizontal or vertical. This interaction and the interaction between DIMEN and PLOT are described in Section 6.4.7 in the documentation on PLOT. The SPACE parameter determines the distance between the startpoint of one PLOT segment and the startpoint of the next PLOT segment. In the case of filled PLOTS, the START argument

specifies the point of the PLOT that is closest to the baseline (see Figure 7). The baseline, in the case of a filled PLOT, is determined by the BASE argument. In the case of a line PLOT, the BASE argument should have a value of zero. See Section 6.5.11 on the BASE argument for a more detailed discussion.

Figure 8 shows an example of a line PLOT that is described by the following instruction.

```
      PLOT START = 450, 120; BASE = 0; SCAN = 4;
1         POINT = 470, 460, 465, 475, 455; DIMEN =
2         12, 9; SPACE = 6, 11
```

INTERACTION OF THE START ARGUMENT AND FILLED PLOTS



RIGEL Code:

```
(A)   PLOT START = 150,100; BASE = 160;
    1   SCAN = 4;  DIMEN = 7,9;  SPACE = 7,9;
    2   POINT = 130,130,130
      PLOT START = 170, 100; POINT = 190,190,190

(B)   PLOT START = 480,100;  BASE = 480;
    1   SCAN = 4;  DIMEN = 7,9;  SPACE = 7,9;
    2   POINT = 460, 460. 460
      PLOT POINT = 500,500,500
```

Figure 7

A LINE PLOT

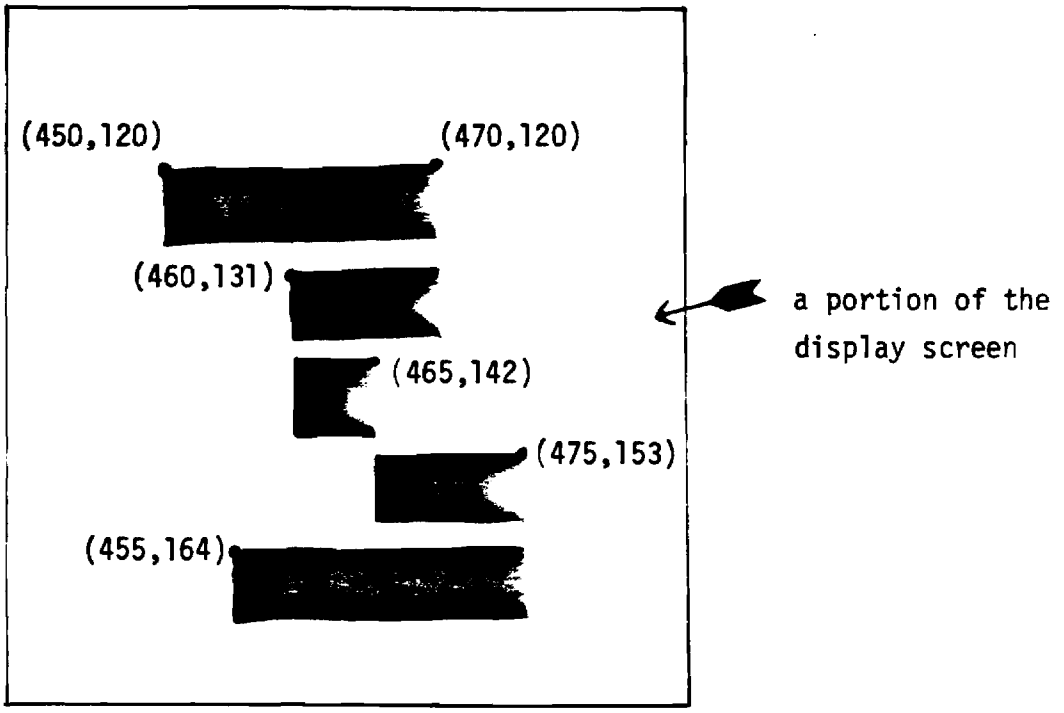


Figure 8

6.7.6 INTERACTIONS CONCERNING FONT

The DIMEN argument defines the size of the FONT character. The DIMEN width value can be from 1 to 8 pixels and the height value can be from 1 to 12 pixels. If the DIMEN values are less than the FONT character size, the right and bottom portions of the character are not displayed.

The START value should be the top left corner of the FONT character. The twelve bytes (6 words) of FONT-definition data define which pixels receive the FOREGD or BACKGD value and which pixels receive no value. The "one" bits in the data cause the respective pixels to receive FOREGD or BACKGD values, depending upon the BK value, and the "zero" bits cause the respective pixels to receive no values. The high-order bit of each FONT data byte represents the left margin of the FONT character matrix and the low-order bit of each FONT data byte represents the right margin of the FONT character matrix.

Since the TEXT instruction is used to display a FONT character, the arguments and modes that affect TEXT also affect the FONT characters when they are displayed.

Two examples of FONT code definitions are:

```
FONT CODE = 101;  ARRAY = IPI, 6
FONT CODE = 102;  ARRAY = IBLNK, 6
```

where the arrays IPI and IBLNK have been previously defined. The following statements would make use of these FONT definitions:

```
IDUM (1) = 101 * 256 + 102
TEXT  ARRAY = IDUM, 1
```

Note that this format requires two FONT characters per array.

7.0 USER'S SYMBOL LIBRARY

A very useful feature of RIGEL is the ability to create and use customized user symbols, such as electronic and mechanical symbols, and general symbols, such as circles. Note that this is different from the FONT specification provided by Ramtek, which forces the user to define a symbol pixel by pixel in at most an 8 x 12 format. These user symbols, once defined, can then be located at any desired point on the screen, rotated to various orientations, and displayed in any of the eight available colors. General symbols can also be scaled to any size.

Symbols are defined in SYMDEF programs (see Sections 6.2, 6.3.3, 6.3.6, and 6.3.7 above for details). A user symbol is then implemented in the user's program by a statement of the form:

SYMBOL symname arg1; arg2; ... argn

where "symname" is the name of the user symbol, and "argi" is either a Ramtek argument (see Section 6.5), a Ramtek mode (see Section 6.6), or a symbol argument. There are currently ten special arguments which apply only to user symbols. These are:

(1) - (2) Coordinates which specify the location of the symbol in x,y pixel notation:

POSX = x

POSY = y

where "x" is a value between 0 and 639, and "y" is a value between 0 and 255; "x" and "y" may be valid integer expressions or variables;

(3) rotation, specified in degrees:

ROTATE = n

where "n" is a valid integer expression, variable, or constant whose value equals 0, 90, 180, or 270;

(4) size of the symbol:

SIZE = n

where "n" is a valid integer expression, variable, or constant; in general, the size is a measurement of the width, height, or radius of the symbol, measured in pixels;

(5) color of the symbol:

COLOR = hue

where "hue" is any of the eight available Ramtek colors (RED, GREEN, BLUE, WHITE, BLACK, MAGENT, CYAN, and YELLOW), or NONE (to produce a black outline).

(6) - (10) five general arguments available to the user for any purpose:

SARG1, SARG2, SARG3, SARG4, SARG5.

Appendix D portrays the symbols which are currently available and illustrates how these arguments apply to them.

8.0 RESTRICTIONS

RIGEL was designed as a one-pass translator, in order to conserve core and execution time. Because of this, the RIGEL programmer must observe the following restrictions upon the FORTRAN statements in his program:

(1) RIGEL must invent statement numbers in the process of converting the user's graphics commands to FORTRAN. It does so by beginning with the largest integer, 32767, and generating successively smaller numbers as it needs them. Do not use a number which will be generated by the translator. A good rule of thumb is to avoid using statement numbers larger than 30000.

(2) Blanks are meaningful separators in RIGEL. Use them to separate different words, but do not use them in the middle of key words.

(3) One graphics statement in the user's program may expand into more than one FORTRAN statement. If the graphics statement has a statement number, then RIGEL places it on the first line of the corresponding FORTRAN expansion. This is fine for statements referenced by a GO TO, but will produce incorrect results if the statement is the final one of a DO-loop. Therefore, the final statement of DO-loops should always be a FORTRAN statement. A safe and practical scheme is to always terminate DO-loops with CONTINUE statements.

(4) RIGEL key words are reserved and may not be used as variable names. In addition, the names of variables and arrays in the

RIGEL commons are reserved. The names which a user should avoid, except in statements where they satisfy the syntax of RIGEL, are:

BACKGD	ICAN	IRAMBF	NFCODE	SIZE
BASE	ICOMP	IRAMCR	NMODUL	SPACE
BK	IDATA	IRAMPR	NONE	START
(B)BLK	IDEBUG	IUFT	NWDARG	(B)WHT
(B)BLU	IDISCR	IX	NWDATA	WINDOW
COEFF	IENDPG	IX1	POSX	(BL)YEL
COLOR	IFDATA	IX2	POSY	
	IHILIM	LDATA	(BL)RED	
DIMEN	INDARG	LIMCHK	ROTATE	
FOREGD	INSTRC	LOCARG	SARG1	
(BL)GRN	IOPCD	LOWLIM	SARG2	
IARGNO	IOPCOD	LUNDIS	SARG3	
IARGS	IOPFLG	LUNLST	SARG4	
IARGVL	IPICNO		SARG5	
IBK	IPROG	MAXBUF	SCALE	
IBUFWD	IPTRIB	NDEVIC	SCAN	

In addition to these restrictions upon the FORTRAN statements in a user's display program, there are four simple rules which he must observe in creating his program. These items are explained elsewhere in this documentation under the appropriate subject headings, but they are also listed here for convenience:

1. The first line of a program must be "FIXED", "RTIME", or "SYMDEF." If the analyzer cannot find one of these three key words, it is unable to proceed any further. Almost any other error can be detected, and program analysis will proceed, but this error is severe.
2. The user must include RIGEL's set of commons in his program. An easy method for doing this is outlined in Section 10.2.1.
3. Any FIXED or RTIME program must include one BEGPIC and at least one ENDPIC statement. BEGPIC causes initialization of the RIGEL variables to occur, and ENDPIC causes the (remaining) contents of the display buffer to be transmitted to the Ramtek.
4. Symbols may not be used to define other symbols. For example, line (3) below is illegal:

```
SYMDEF          (1)
DEFINE SYM1     (2)
. . .
SYMBOL SYMX           (3)
. . .
ENDDEF          (4)
```

9.0 ERROR HANDLING

The portion of RIGEL which parses the user`s program and translates it into FORTRAN produces three types of error messages(see Appendix E for a listing of the messages):

(1) "user errors" occur when the user violates the syntax of the RIGEL language, but not so badly that the preprocessor cannot continue to analyze the remainder of the instructions in his program. This should be the type of error message encountered most frequently by the user, as when he misspells the name of an instruction or argument, forgets to specify data for an instruction, or does not supply a separating semi-colon between arguments.

(2) "fatal user errors" occur when the user does something so drastic that RIGEL cannot decide how to analyze the rest of his program. Rather than waste computer time and list a lot of potentially meaningless error messages, it simply terminates analysis of the user`s program. Two typical sources of such a message would be if the user forgot to specify one of the key words "FIXED", "RTIME", or "SYMDEF" as the first instruction, or if he accidentally specified an input file which was empty or contained garbage. This message is also generated if the user exceeds the generous number of errors which are permitted in one program -- quite likely, he may have a repetitive error of some sort.

(3) "fatal RIGEL errors" hopefully should never be seen by the user. They indicate a serious problem with RIGEL, rather

than with the user's program. The designers of RIGEL should be notified if such an error does occur, and listings of input and output should be saved for their reference.

As RIGEL analyzes the user's program, it generates a listing file of the source on the listing device. If an error is detected on a line, an asterisk is placed beneath the character where the error was detected, and then an error message is printed, such as:

```
VECTOR STRRT = 10, 50      (where START is misspelled)
             *
USER ERROR
ARGUMENT NAME IS INVALID
```

NOTE: The user should never try to compile and execute a display program if RIGEL detected any errors of any type in it.

Graphics Assembler Messages

If the user specifies any argument values that are greater than the RIGEL upper limit or less than the RIGEL lower limit for that argument, then the Graphics Assembler modules reset these values to the upper limit or the lower limit, respectively, and print a warning message in the Ramtek output listing. However, if after indexing, the argument values exceed the screen resolution, then indeterminate data may result.

10.0 PROCEDURE FOR CREATION AND EXECUTION OF A DISPLAY

Please note that details of this section pertain directly to using RIGEL on a Digital Equipment Corporation PDP-11/70 computer with RSX-11/M Operating System and FORTRAN F4P. There are several ways to set up procedures that would accomplish the various steps necessary to generate a display; we will simply outline one method which we found to be convenient and make general recommendations about how to proceed.

Figure 3 provides an overview of the process of creating a display on a Conrac Monitor using the RIGEL Graphics Preprocessor and the RIGEL graphics generation routines. While RIGEL functions in two parts, namely:

1. preprocessing (conversion to FORTRAN) and
2. linking with the graphics subroutines, several steps are necessary to accomplish the complete task. These include:
    1. Creation of the user's source program.
    2. Preprocessing by the RIGEL Preprocessor of the source program to produce a FORTRAN program.
    3. a) If the source program is a FIXED or RTIME program, it is compiled, linked with the graphics assembler modules and the user symbol library, and stored as an executable task.
    b) If the source program is a SYMDEF program, it is compiled and added to the symbol library.

4. When the executable task created above (step 3.a).) is executed, and the user specifies which of the three Conrac monitors to use, the Ramtek object code for the picture is generated and sent to the specified monitor(s).

10.1 FILES AND CONVENTIONS

It is assumed that the user is familiar with the RSX-11/M Operating System. In our implementation of RIGEL, we use the following standard files:

FILE	USE
RIGEL.FTN	RIGEL preprocessor source file
RIGEL.OBJ	compiled version of RIGEL.FTN
RIGEL.TSK	RIGEL preprocessor executable task
RIGINP.FTN	user's input file for preprocessing
RIGLST.LST	listing of user's input file and error messages
RIGFTN.FTN	output FORTRAN source file from preprocessor
RIGFTN.OBJ	compiled version of RIGFTN.FTN
RIGASL.FTN	RIGEL graphics assembler subroutines (source file)
RIGASL.OBJ	compiled version of RIGASL.FTN
RIGFTN.TSK	executable task formed by linking RIGFTN.OBJ with RIGASL.OBJ
SOURCE.FTN	shell of an input file for editing purposes

10.2 SPECIAL PROCEDURES

If the user wishes, he can establish procedures to simplify the process of creating a display. We shall outline briefly the files which are used and describe the steps which are necessary to generate a picture.

10.2.1 CREATION OF THE USER'S SOURCE PROGRAM

The input source program is created with the Text Editor. Since the RIGEL common blocks and equivalence and data statements need to be included in the user's source program, we have found it useful to maintain a file named SOURCE.FTN with these items in it. SOURCE.FTN is simply the shell of a user's program. SOURCE.FTN is edited, and the output file is stored under another filename.

10.2.2 PREPROCESSING BY RIGEL

The RIGEL Preprocessor expects the user's source file to be in RIGINP.FTN, so the file created in 10.2.1 should be copied into RIGINP.FTN. The Preprocessor generates the FORTRAN translation of the program (RIGFTN.FTN) and a listing of the input (including error messages, if any, in RIGLST.LST). THE USER SHOULD NEVER ATTEMPT TO ASSEMBLE OR EXECUTE A PICTURE PROGRAM IF RIGEL INDICATES AT THIS POINT THAT IT CONTAINS ANY ERRORS.

10.2.2.1 PREPROCESSING OF A FIXED OR RTIME PROGRAM

File RIGINP.FTN is compiled, and file RIGFTN.OBJ is linked with the compiled version of the Graphics Assembler Routines (RIGASL.OBJ) to form the executable task RIGFTN.TSK (which can be stored under a different name, if desired).

10.2.2.2 PREPROCESSING OF A SYMDEF SYMBOL PROGRAM

File RIGINP.FTN is compiled, and file RIGINP.OBJ is added to the user's library of symbols.

10.2.3 EXECUTION OF THE PICTURE PROGRAM

The RIGFTN.TSK file created above is executed. The user must input a Conrac screen number (1, 2, or 3) for an RTIME program.

APPENDIX A A1

RIGEL SUMMARY SHEET

PART I — CONTROL STATEMENTS

FIXED
RTIME } key word to define program type; is first word of program
SYMDEF

BEGPIC } used in FIXED and RTIME programs to initialize and terminate graphics
ENDPIC

DEFINE symname } used in SYMDEF programs to initialize and terminate symbol
ENDDEF } definitions

SYMBOL symname } used in FIXED and RTIME programs to reference symbols in the
} user's symbol library

PART II — INSTRUCTIONS & THEIR ASSOCIATED DATA

INSTRUCTION	PERMISSIBLE DATA ARGUMENTS
SET	(no data arguments)
ERASE	(no data arguments)
TEXT	ARRAY = name, length or STRING = "abc" (or 'abc') or VALUE = xx.x, m, n or VALUEL = xx.x, m, n
RASTER	ARRAY = name, length
VECTOR	END = x1, y1; END = x2, y2; ... END = xn, yn or ARRAY = name1, name2, length
CONIC	END = x1, y1; END = x2, y2; ... END = xn, yn or ARRAY = name1, name2, length
PLOT	POINT = p1, p2, ... pn or ARRAY = name, length
FONT	CODE = n; ARRAY = name, length

PART III — RAMTEK ARGUMENTS

FOREGD = color *

BACKGD = color *

IX1 = x, y

IX2 = x, y

WINDOW = X1, Yt, Xr, Yb

SCAN = m ($0 \leq m \leq 7$)

DIMEN = width, height

SPACE = horizontal, vertical

SCALE = m

COEFF = A, B, C, D, E, K

BASE = x (or BASE = y)

START = x, y

*where: color can be any of the following capitalized items

	full intensity	2/3 intensity	blinking
red	RED	LRED	BRED, BLRED
green	GRN	LGRN	BGRN, BLGRN
blue	BLU		BBLU
yellow		LYEL	BLYEL
white	WHT		BWHT
black	BLK		BBLK

PART IV -- RAMTEK MODES

IX = n	($0 \leq n \leq 3$)	[addressing]
BK = n	(n = 0 or 1)	[reverse background]

PART V -- USER SYMBOL ARGUMENTS

ROTATE = n (n = 0, 90, 180, or 270)

SIZE = n

POSX = x

POSY = y

COLOR = color

SARG1 = n

SARG2 = n

SARG3 = n

SARG4 = n

SARG5 = n

APPENDIX B

ARGUMENT VALUE TABLES

PART I -- INTERACTION OF COP, WINDOW, AND SCAN

SCAN	0	1	2	3	4	5	6	7
COP x-value	Xl	Xr	Xl	Xr	Xl	Xl	Xr	Xr
COP y-value	Yt	Yt	Yb	Yb	Yt	Yb	Yt	Yb

PART II -- SCAN DIRECTIONS FOR RASTER INSTRUCTION

SCAN	0	1	2	3	4	5	6	7
PRIMARY DIRECTION	→	←	→	←	↓	↑	↓	↑
SECONDARY DIRECTION	↓	↓	↑	↑	→	→	←	←

PART III -- SCAN DIRECTIONS FOR TEXT INSTRUCTION

SCAN	0	1	2	3	4	5	6	7
PRIMARY DIRECTION	H	H	H	H	V	V	V	V
SECONDARY DIRECTION	V	V	V	V	H	H	H	H

where: H = horizontal direction

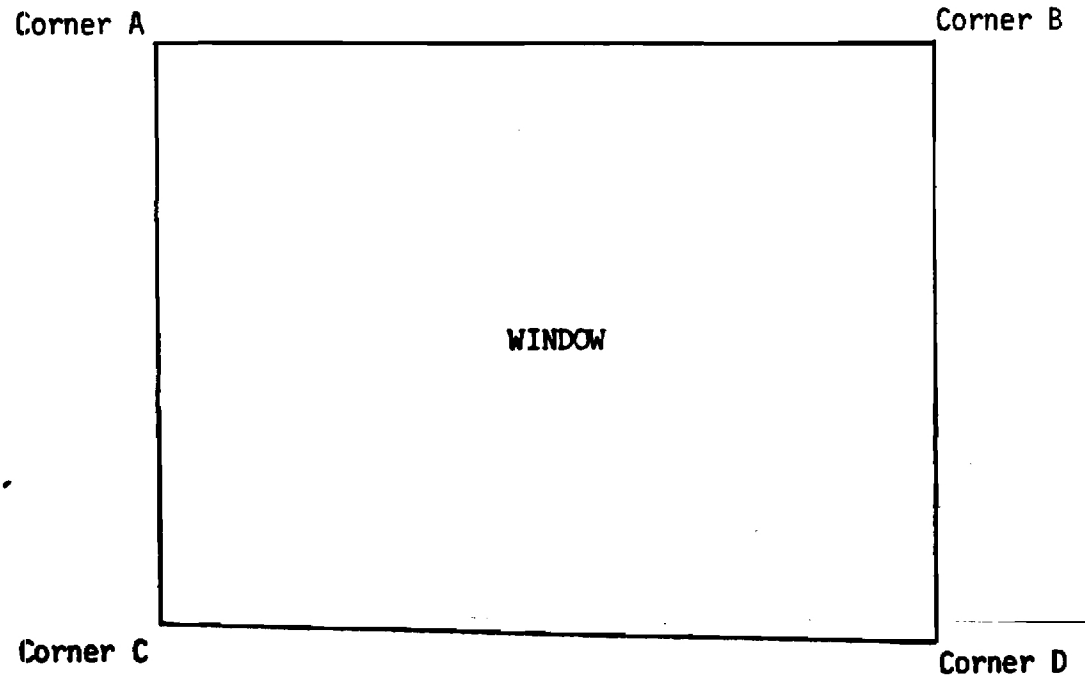
V = vertical direction

PART IV — SCALE VALUES

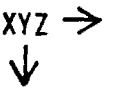
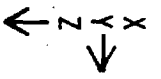
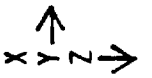
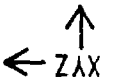
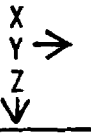
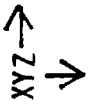
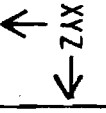
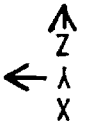
Scale Value	Y-Scale, X-Scale	PICTURE ELEMENTS (Displayed: Received) Y-Ratio, X-Ratio
-22	-2 , -2	1:4 , 1:4
-21	-2 , -1	1:4 , 1:2
-20	-2 , 0	1:4 , 1:1
-12	-1 , -2	1:2 , 1:4
-11	-1 , -1	1:2 , 1:2
-10	-1 , 0	1:2 , 1:1
-2	0 , -2	1:1 , 1:4
-1	0 , -1	1:1 , 1:2
0	0 , 0	1:1 , 1:1
1	0 , 1	1:1 , 2:1
2	0 , 2	1:1 , 4:1
10	1 , 0	2:1 , 1:1
11	1 , 1	2:1 , 2:1
12	1 , 2	2:1 , 4:1
20	2 , 0	4:1 , 1:1
21	2 , 1	4:1 , 2:1
22	2 , 2	4:1 , 4:1

PART V -- INTERACTION OF START, WINDOW, AND SCAN

SCAN	0	1	2	3	4	5	6	7
WINDOW ORIGIN	A	B	C	D	A	C	B	D



PART VI -- SCAN AND SPACE VALUES

SCAN VALUE	SPACE: X-VALUE	SPACE: Y-VALUE	SAMPLE DISPLAY
0	W	H	
1	-H	W	
2	H	-W	
3	-W	-H	
4	W	H	
5	H	-W	
6	-H	W	
7	-W	-H	

where: W = Character Width

H = Character Height

APPENDIX C

DECLARATIONS REQUIRED BY RIGEL

```
C*
C   INTRODUCE NAMES FOR COLORS AND ARGUMENTS
C
      INTEGER BLK, LRED, LGRN, LYEL, GRN, RED, BLU, WHT,
     1    BBLK, BLRED, BLGRN, BLYEL, BGRN, BRED, BBLU, BWHT
      INTEGER FOREGD, BACKGD, IX1(2), IX2(2), WINDOW(4), SCAN, DIMEN(2),
     1    SPACE(2), SCALE, COEFF(6), BASE, START(2), IX, BK, POSX, POSY,
     2    ROTATE, SIZE, COLOR, SARG1, SARG2, SARG3, SARG4, SARG5

C
C   INSERT GRAPHICS PREPROCESSOR COMMONS
C
      COMMON /RIGCOM/  IPROG,IPICNO,LUNLST,LUNDIS,NDEVIC
      COMMON /RAMFIX/  IARGVL(16),NWDARG(16),LOCARG(16),LIMCHK(16),
     1    LOWLIM(30),IHILIM(30),IDISCR(33),IOPCD(8),IFDATA(8),
     2    NMODUL(3,10)
      COMMON /RAMTEK/  IRAMCR(30),IRAMPR(30),INDARG(16),IDATA(200),
     1    IARGS(10)
      COMMON /CGRIN/ IARGNO,IBK,IBUFWD,ICOMP,IDEBUG,IENDPG,INSTRC,
     1    IOPCOD,IOPFLG,IPTRIB,IX,LDATA,MAXBUF,NFCODE,NWDATA
      COMMON /DISBUF/ IRAMBF(500)

C
C   EQUIVALENCE NAMES OF ARGUMENTS TO THEIR CORRECT POSITIONS
C     IN THE CORRESPONDING BUFFERS
C

      EQUIVALENCE (FOREGD,IRAMCR(2)),(BACKGD,IRAMCR(3)),
     1    (IX1(1),IRAMCR(4)),(IX2(1),IRAMCR(6)),(WINDOW(1),IRAMCR(10)),
     2    (SCAN,IRAMCR(14)),(DIMEN(1),IRAMCR(15)),(SPACE(1),IRAMCR(17)),
     3    (COEFF(1),IRAMCR(21)),(BASE,IRAMCR(27)),(START(1),IRAMCR(29)),
     4    (SCALE,IRAMCR(19))
      EQUIVALENCE (BK,IBK)
      EQUIVALENCE (POSX,IARGS(1)),(POSY,IARGS(2)),(ROTATE,IARGS(3)),
     1    (SIZE,IARGS(4)),(COLOR,IARGS(5)),(SARG1,IARGS(6)),(SARG2,IARGS(7)),
     2    (SARG3,IARGS(8)),(SARG4,IARGS(9)),(SARG5,IARGS(10))

C
C   DEFINE VALUES FOR COLORS
C

      DATA BLK, LRED, LGRN, LYEL, GRN, RED, BLU, WHT,
     1    BBLK, BLRED, BLGRN, BLYEL, BLGRN, BLRED, BBLU, BWHT
     2    /0, "10421, "21042, "31463, "42104, "52525, "63146, "73567,
     3    "104210, "114631, "125252, "135673, "146314, "156735,
     4    ""167356, "177777/
C**
```

APPENDIX D

CURRENTLY IMPLEMENTED USER SYMBOLS

In Parts I and II of this Appendix, diagrams are given which represent the Mechanical, Electrical, and General Symbols as they are currently implemented. The relationship between the symbol arguments and the symbols is as follows:

1. The point specified by (POSX, POSY) is the starting location of the symbol. This point is identified by a . on each diagram. After the symbol is drawn, a new (POSX,POSY) location is calculated. This point is identified by a x on each diagram and represents the default starting location of the next symbol.
2. ROTATE specifies one of the four orientations -- 0, 90, 180, or 270 degrees. Note that each symbol is rotated counterclockwise from its 0 degree orientation.
3. SIZE is used only for General Symbols. SIZE specifies the horizontal radius in pixels of CIRCLE and SEMICR. SIZE should be an odd number, in order to produce a symmetric symbol. If SIZE is even, RIGEL will add one pixel. SIZE is indicated by S on each applicable diagram.
4. COLOR, which can be any of the eight permissible Ramtek hues, specifies the internal shading color of each symbol (except for COIL and RESIST). If the user specifies COLOR = NONE, only the black outline of each figure is produced.
5. SARG1 and SARG2 are used when two size dimensions are needed for a General Symbol. SARG1 (indicated by A) is horizontal length in pixels, and SARG2 (indicated by A) is vertical height in pixels. Like SIZE, SARG1 and SARG2 should be odd numbers; even values will be incremented by one pixel by RIGEL.

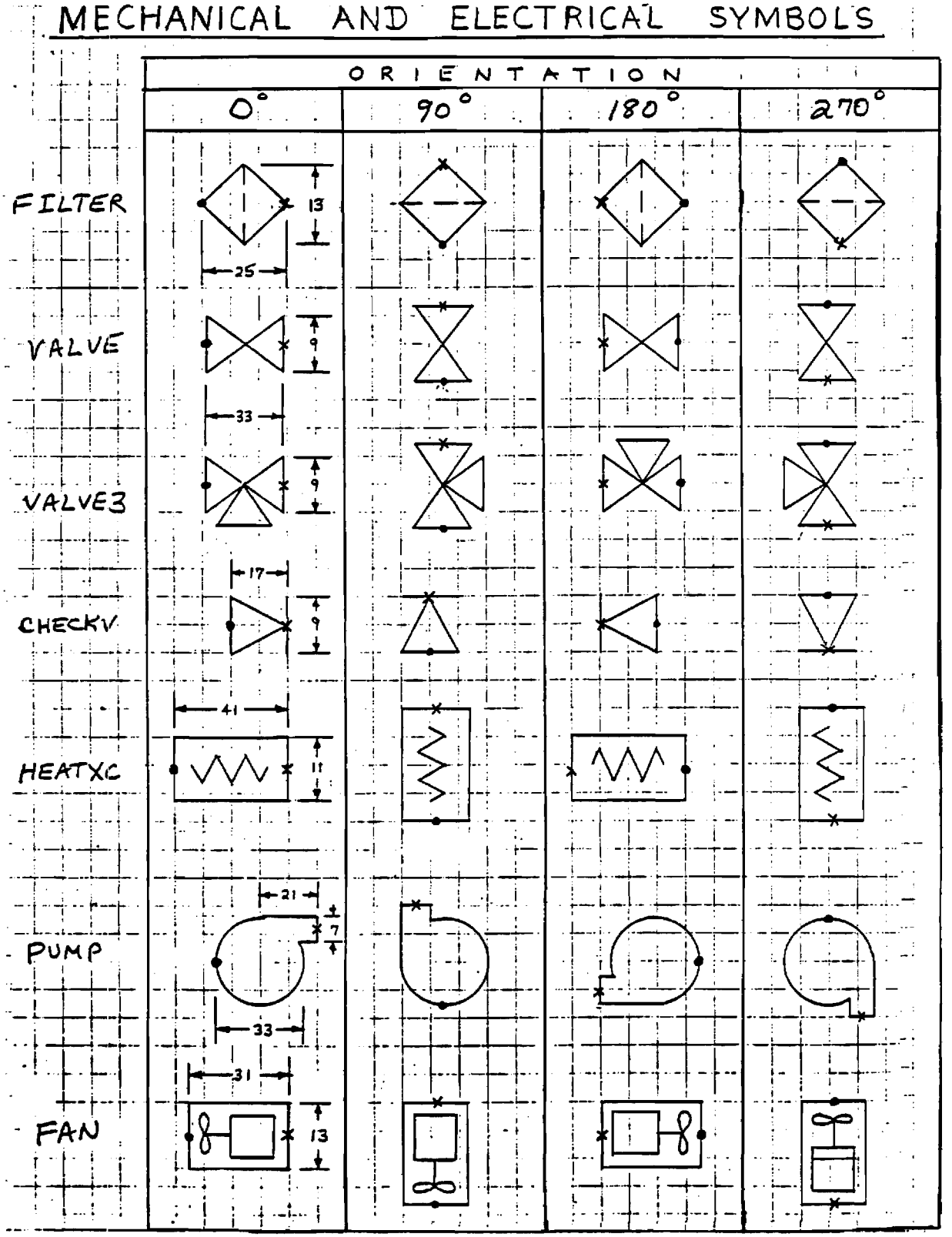


Figure reproduced with the permission of Mr. Robert Stoakley, Cadre Corp.

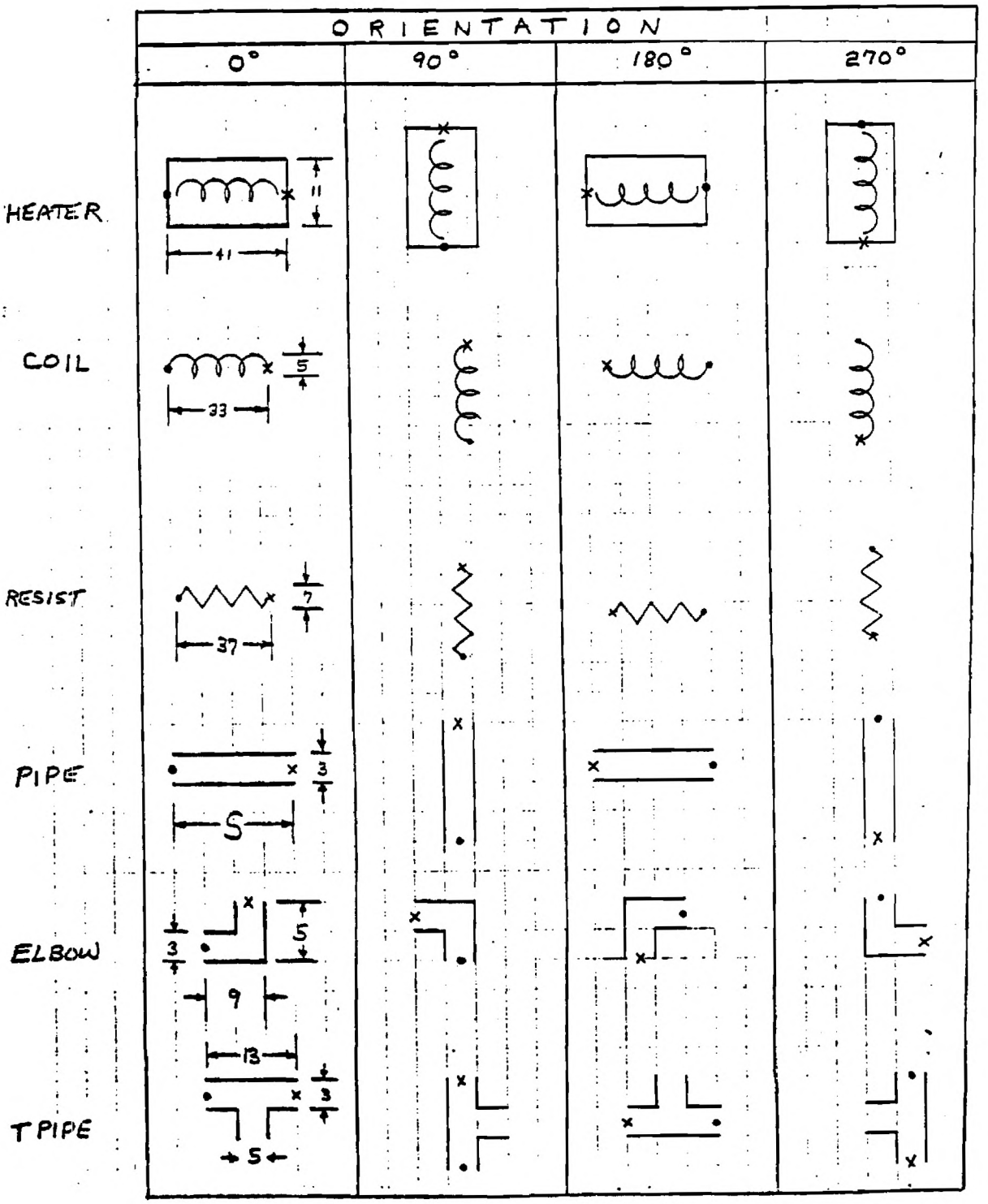


Figure reproduced with the permission of Mr. Robert Stoakley, Cadre Corp.

PART II -- GENERAL SYMBOLS

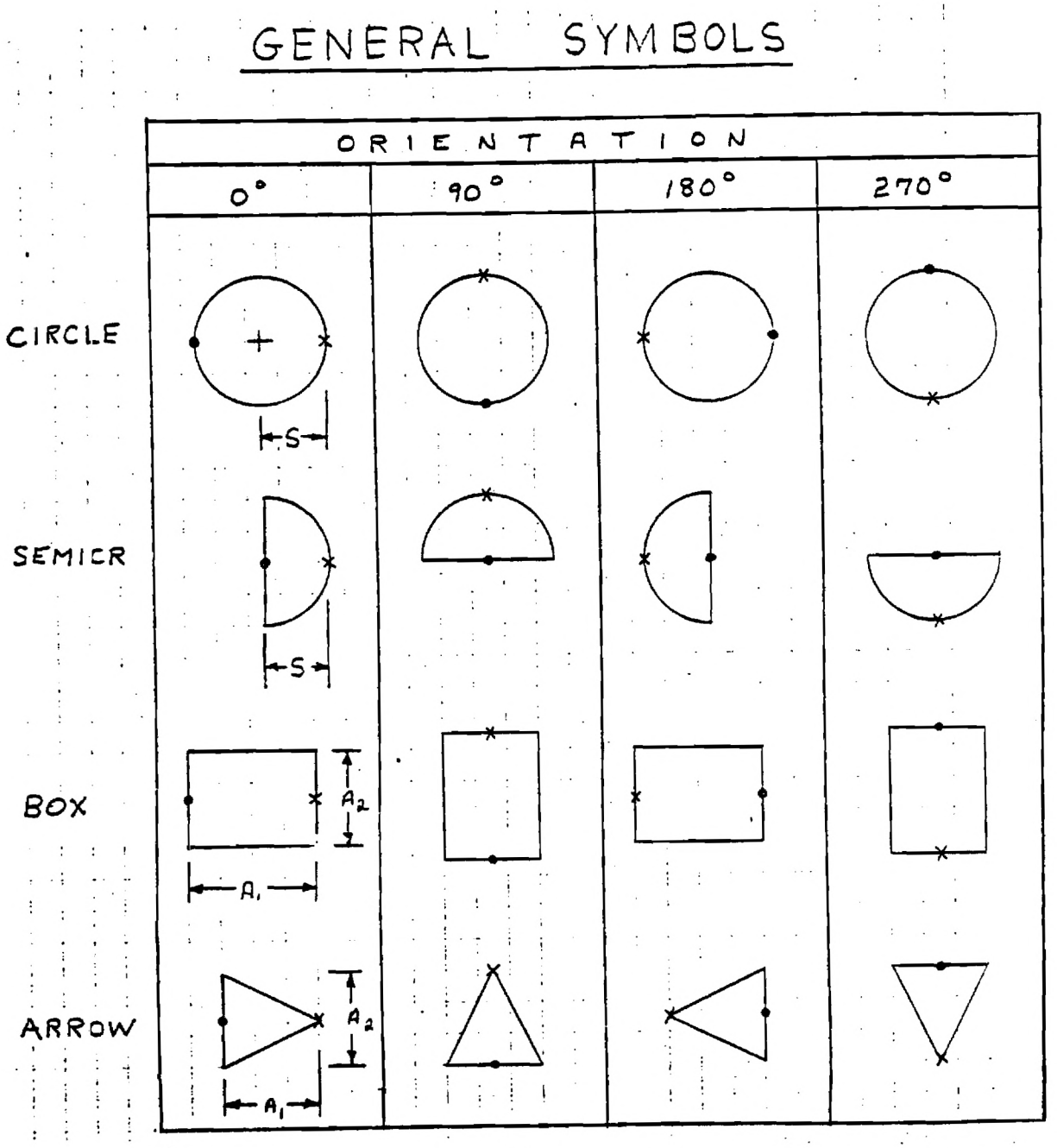


Figure reproduced with the permission of Mr. Robert Stoakley, Cadre Corp.

APPENDIX E

RIGEL ERROR MESSAGES

PART I -- USER ERRORS

1. The first word of the program cannot be followed by a continuation line.
2. Arguments or data values are not permitted for this instruction.
3. The graphics initialization flag cannot be set more than once.
4. Symbol name is missing or invalid.
5. Symbol name must be followed by a space or end-of-statement.
6. Symbol name must be followed by an end-of-statement.
7. Every "DEFINE" statement must be followed by an "ENDDEF" statement before another "DEFINE" statement is used.
8. Each "ENDDEF" statement must be preceded somewhere by its own "DEFINE" statement.
9. A macro symbol instruction cannot be nested inside a macro symbol definition.
10. A macro symbol can only be defined by a SYMDEF program.
11. A remote variable can only be used in a Real-Time program.
12. The graphics routines cannot be initialized from inside a Symbol Definition program.
13. The graphics routines cannot be terminated from inside a Symbol Definition program.
14. Argument or data name must be followed by an equal sign.
15. Argument name is invalid.
16. Argument or data name is invalid.
17. Instruction lacks data.
18. At least one argument value is missing.
19. Argument values must be separated by a comma.
20. Argument strings must be separated by a semi-colon.
21. Argument value cannot be a quotation.

22. Argument value is missing.

23. Mode value is missing.

24. Array name is missing or invalid.

25. Number of words in array is missing or invalid.

26. Argument value must be a quotation.

27. X value is missing or invalid.

28. Y value is missing or invalid.

29. Point value is missing or invalid.

30. Font code value is missing or invalid.

31. Data value is missing or invalid.

32. Format is missing or invalid.

33. Arithmetic expression has unmatched parentheses.

PART II -- FATAL USER ERRORS

1. The first word of the program must be FIXED, RTIME, or SYMDEF.
2. Input source file is empty.
3. Source statements contain errors - therefore, no picture can be created.
4. A FIXED program must contain a BEGPIC statement.
5. A FIXED program must contain at least one ENDPIC statement.
6. A REAL-TIME program must contain a BEGPIC statement.
7. A REAL-TIME program must contain at least one ENDPIC statement.
8. Source statements contain errors - therefore, no macro symbol subroutine can be created.
9. "DEFINE" and "ENDDEF" statements have not been properly placed in the program.
10. Fatal user error in ERROUT — Number of user errors has exceeded the maximum number of allowable errors.

PART III -- FATAL RIGEL ERRORS

1. Program type (IPROG) has assumed an invalid value.
2. Statement type (ISTYPE) has assumed an invalid value.
3. Data indicator (IARGNO) has assumed an invalid value.

PART IV -- WARNING MESSAGES

1. Warning -- Argument no. (IARGNO) has value of (ICURNT) - is out of range. Value set to (LOWLIM (LOCATN)).

2. Warning -- Argument no. (IARGNO) has value of (ICURNT) - is out of range. Value set to (IHILIM (LOCATN)).

3. Warning -- Argument no. (IARGNO) has invalid value of (ICURNT). Color will not be changed.

4. Warning -- Invalid total number of digits (I) specified for text value. Number set to 10.

5. Warning -- Invalid number of decimal fraction digits (J) specified for text value. Number set to zero.

6. Warning -- Magnitude of real number (R) cannot be represented by value format specification (I,J).

7. Fatal Graphics Assembler error in COPCOD — invalid instruction number = (INSTRC).